Document Title	Specification of Manifest
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	713

Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R20-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2020-11-30	R20-11	AUTOSAR Release Management	• Remodeling of Phm contribution • Reporting of Security Events • Support for cryptographic Operations • Remodeling of Diagnostic Mapping • minor corrections / clarifications / editorial changes
2019-11-28	R19-11	AUTOSAR Release Management	• Overhaul of Signal-to-Service Translation • Support for Raw Data Streams • Support for Vehicle Package • Support for Service Versioning • Changed Document Status from Final to published
2019-03-29	19-03	AUTOSAR Release Management	• Introduction of Diagnostic Port Interfaces • Overhaul of Software Cluster and introduction of Software Package • Support for Identity and Access Management • Network Management Configuration
2018-10-31	18-10	AUTOSAR Release Management	• Finish introduction of CppImplementationDataType • Support for optional elements in structures • Rework configuration of adaptive platform modules

2018-03-29	18-03	AUTOSAR Release Management	• Time Synchronization • DDS Deployment
2017-10-27	17-10	AUTOSAR Release Management	• Optional elements in Service Interfaces • Interaction with web services • Secure Communication • Support for interaction with crypto and persistency • Signal-to-Service translation • Support for E2E communication • Platform Health Management • Uploadable Software Package
2017-03-31	17-03	AUTOSAR Release Management	• Initial release

Disclaimer

Table of Contents

1 Introduction 17

1.1 Modeling Approach . . . 18
1.2 The Term Service . . . 19
1.3 Abbreviations . . . 19
1.4 Document Conventions . . . 21
1.5 Requirements Tracing . . . 24
1.6 Known Limitations . . . 30

2 Big Picture of Manifest Definition 31

2.1 Design vs. Deployment . . . 31
2.1.1 Overview . . . 31
2.1.2 Relation between Design and Deployment Models . . . 31
2.1.3 Structure of the document . . . 32
2.2 About Manifest . . . 32
2.3 Serialization Format . . . 32
2.4 Scope . . . 33
2.5 Manifests described in this Document . . . 34

3 Application Design 36

3.1 Overview . . . 36
3.2 Software Component . . . 36
3.3 Data Type . . . 38
3.3.1 Overview . . . 38
3.3.2 ApplicationDataType . . . 38
3.3.2.1 String Data Type . . . 39
3.3.2.2 Associative Map Data Type . . . 41
3.3.2.3 Attributes of SwDataDefProps . . . 46
3.3.3 CppImplementationDataType . . . 48
3.3.3.1 Overview . . . 48
3.3.3.2 Attributes of SwDataDefProps . . . 61
3.3.3.3 Primitive Data Types . . . 62
3.3.3.4 String Data Type . . . 63
3.3.3.5 Array Data Type . . . 64
3.3.3.6 Vector Data Type . . . 66
3.3.3.7 Struct Data Type . . . 69
3.3.3.8 Enumeration Data Type . . . 70
3.3.3.9 Map Data Type . . . 72
3.3.3.10 Variant Data Type . . . 73
3.3.3.11 Bitfield Data Type . . . 74
3.3.4 Compatibility of ApplicationDataType and CppImplementationDataType . . . 75
3.4 Service Interface . . . 77
3.4.1 Overview . . . 77
3.4.2 Event . . . 80

3.4.3 Field . . . 80
3.4.4 Method . . . 82
3.4.4.1 Fire and Forget Method . . . 83
3.4.5 Versioning of ServiceInterfaces . . . 84
3.4.5.1 Versioning driven by transport layer . . . 85
3.4.6 Namespace . . . 86
3.4.7 Error Handling . . . 89
3.4.8 Service Interface Data Type Mapping . . . 92
3.4.9 Communication Group pattern . . . 95
3.5 Service Interface Mapping . . . 96
3.6 Service Interface Element Mapping . . . 101
3.6.1 Overview . . . 101
3.6.2 Service Interface Event Mapping . . . 104
3.6.3 Service Interface Field Mapping . . . 105
3.6.4 Service Interface Method Mapping . . . 107
3.7 Service Needs . . . 108
3.7.1 Overview . . . 108
3.7.2 Service Needs for Diagnostics . . . 108
3.8 Persistency Interface . . . 111
3.8.1 Overview . . . 111
3.8.1.1 The big Picture . . . 111
3.8.1.2 Modeling of Persistency Interface . . . 112
3.8.1.3 Redundancy Handling . . . 114
3.8.1.4 Update Handling . . . 118
3.8.2 Persistency Key Value Storage Interface . . . 119
3.8.3 Persistency File Storage Interface . . . 121
3.9 Time Synchronization Interface . . . 122
3.10 Platform Health Management Interface . . . 125
3.10.1 Overview . . . 125
3.10.2 Supervised Entities and Checkpoints . . . 126
3.10.3 Health Channels . . . 128
3.10.4 Recovery notification to State Management . . . 130
3.11 Diagnostic Interface . . . 133
3.11.1 Overview . . . 133
3.11.2 Diagnostic Routine Interface . . . 135
3.11.3 Interface to Data Identifier and Element of Data Identifier . . . 138
3.11.4 Interface to diagnostic Events . . . 141
3.11.5 Interface to diagnostic Condition . . . 142
3.11.6 Indicator Interface . . . 143
3.11.7 Security Level Interface . . . 144
3.11.8 Service Validation Interface . . . 145
3.11.9 Operation Cycle Interface . . . 147
3.11.10 Generic UDS Interface . . . 148
3.11.11 DoIP Interfaces . . . 149
3.11.12 Diagnostic Interfaces for Upload and Download . . . 151
3.11.13 Interface to support managing the EcuReset . . . 153

3.12 Crypto Interfaces . . . 154
3.12.1 Interaction with Crypto Software . . . 154
3.12.2 Crypto Key Slot Interface . . . 155
3.12.3 Crypto Certificate Interface . . . 159
3.12.4 Crypto Provider Interface . . . 160
3.12.5 Crypto TrustMaster Interface . . . 160
3.12.6 Linking of Crypto Certificate to a Crypto Key Slot . . . 161
3.13 Raw Data Stream Interface . . . 163
3.14 Security Event Report Interface . . . 165
3.15 Interaction Endpoint for Application . . . 167
3.15.1 Service-oriented Communication . . . 167
3.15.2 Interaction with Persistent Key-Value Storage . . . 168
3.15.3 Interaction with Persistent File Storage . . . 169
3.15.4 Port Prototype Props . . . 169
3.15.5 Port Prototype ComSpec . . . 171
3.15.5.1 Port Prototypes typed by Service Interfaces . . . 173
3.15.5.2 Port Prototypes typed by Persistency Data Interfaces 181
3.16 Executable . . . 182
3.17 Optional Members in complex Data Structures . . . 188
3.17.1 Background . . . 188
3.17.2 Definition of Optionality . . . 189
3.18 Serialization Properties . . . 192
3.18.1 Default Values for Serialization Properties . . . 192
3.18.2 Individual Definition of Serialization Properties . . . 198
3.18.3 Assignment of TLV properties . . . 204
3.18.3.1 Assignment of TLV Data IDs . . . 204
3.18.3.2 Assignment of Wire Type Selection . . . 210
3.19 Process Design . . . 210
3.19.1 Deterministic Client Resource . . . 212
3.20 Grant Design . . . 215
3.20.1 Com Grant Design . . . 216
3.20.2 Grant Design for Raw Streaming Data . . . 222
3.20.3 Remote access control . . . 223
3.20.3.1 Remote subject in case of TLS . . . 225
3.20.3.2 Remote subject in case of IPsec . . . 227
3.20.3.3 Remote subject in case of IP communication . . . 229
3.20.3.4 Remote subject in case of SecOC communication . 230

4 Diagnostic Design 231

4.1 Diagnostic Mapping . . . 231
4.1.1 Overview . . . 231
4.1.2 Diagnostic Event to Port Mapping . . . 237
4.1.3 Diagnostic Operation Cycle to Port Mapping . . . 243
4.1.4 Diagnostic Enable Condition to Port Mapping . . . 246
4.1.5 Diagnostic Clear Condition to Port Mapping . . . 248
4.1.6 Diagnostic Indicator to Port Mapping . . . 250

4.1.7 Diagnostic Memory Destination to Port Mapping 252
4.1.8 Diagnostic Security to Port Mapping 254
4.1.9 Diagnostic Data Identifier to Port Mapping 256
4.1.10 Diagnostic Generic UDS Service Handler to Port Mapping . 259
4.1.11 Diagnostic Generic Mapping 260
4.1.12 Diagnostic Data Mapping 261
4.2 Diagnostic Clear Condition 264
4.3 Security Access 266
4.4 DiagnosticProvidedDataMapping 267

5 System Design 269

5.1 Overview 269
5.2 Specification of Communication System Structure 271
5.2.1 Network connection 273
5.2.1.1 Support of 10BASE-T1S Network Topologies 279
5.2.2 TcpIp stack configuration properties 279
5.2.2.1 IP configuration properties 280
5.2.2.2 TCP and UDP configuration properties 286
5.2.2.3 ICMP configuration properties 288
5.2.3 Securing Communication with IPsec 289
5.2.4 Service Discovery Configuration 298
5.2.4.1 SOME/IP Service Discovery Configuration 298
5.2.5 Partial Network 300
5.3 Log and Trace Design 302
5.4 Specification of Application Software System Structure 304
5.5 Modeling of service oriented communication between Classic and Adaptive platform 306
5.5.1 MethodMapping 308
5.5.2 EventMapping 309
5.5.3 FieldMapping 310
5.5.4 FireAndForgetMapping 312

6 Sub-System Design 315

6.1 Overview 315
6.2 Software Cluster Design 315
6.3 Provided and required Services of Software Cluster Design 321
6.4 Mapping of Services to Executables 324

7 Machine Manifest 329

7.1 Process To Machine Mapping 332
7.1.1 General Modeling Approach 332
7.1.2 Core Affinity 334
7.1.3 Default Start-up and Termination Timeout 335

8 Execution Manifest 336

8.1 Overview 336
8.2 Startup Configuration 339

8.2.1 State-dependent Startup Configuration . . . 340
8.2.2 Scheduling . . . 344
8.2.3 Process Arguments . . . 344
8.2.4 Association with Resource Group . . . 345
8.2.5 Execution Dependency . . . 345
8.2.6 Assignment of Processes to Function Group states . . . 348
8.2.7 Resource Consumption Boundaries . . . 348
8.2.8 Error and Termination Behavior . . . 351
8.3 Deterministic Client . . . 352
8.4 Function Groups . . . 353
8.4.1 Semantics of Function Group . . . 353
8.4.2 Machine Function Group . . . 356
8.5 Reporting of Security Events . . . 357
9 Platform Module Development 358
9.1 OS Module configuration . . . 360
9.2 Persistency Deployment . . . 362
9.2.1 Overview . . . 362
9.2.1.1 Redundancy Handling . . . 365
9.2.1.2 Update Handling . . . 365
9.2.1.3 Size Handling . . . 367
9.2.1.4 Security Handling . . . 367
9.2.2 Deployment of Persistent Key-Value Storage . . . 368
9.2.3 Deployment of File Storage . . . 372
9.3 Platform Health Management Deployment . . . 375
9.3.1 Overview . . . 375
9.3.2 Relation between design and deployment . . . 378
9.3.3 Supervision deployment . . . 379
9.3.3.1 `AliveSupervision` definition . . . 383
9.3.3.2 `CheckpointTransition` definition . . . 384
9.3.3.3 `LogicalSupervision` definition . . . 385
9.3.3.4 `DeadlineSupervision` definition . . . 386
9.3.4 Global supervision entity deployment . . . 387
9.3.5 Health channel deployment . . . 389
9.3.5.1 Supervision health channel deployment . . . 390
9.3.5.2 External health channel deployment . . . 391
9.3.6 Recovery Notification . . . 393
9.4 Time Synchronization Deployment . . . 395
9.4.1 Overview . . . 395
9.4.2 Time Synchronization functional cluster configuration . . . 396
9.4.3 Time Base . . . 397
9.4.3.1 Synchronized time base . . . 398
9.4.3.2 Persistent Time Base value storage . . . 401
9.4.3.3 Ethernet synchronized time . . . 402
9.4.4 Time Base to Port Prototype mapping . . . 403
9.5 DoIP configuration . . . 405

9.6 Log and Trace module configuration . . . 411
9.7 Network Management configuration . . . 418
9.7.1 UdpNm configuration constraints . . . 423
9.8 Update and Configuration Management . . . 424
9.9 IAM configuration . . . 426
9.9.1 Com Grant Deployment . . . 427
9.9.2 Grant Deployment for Raw Streaming Data . . . 432
9.9.3 Remote access control . . . 433
9.10 Crypto Deployment . . . 435
9.10.1 Crypto Provider . . . 437
9.10.2 Crypto Key Slot . . . 438
9.10.3 Crypto Certificate . . . 440
9.11 IdsM Deployment . . . 442
9.11.1 IdsM Instantiation . . . 442
9.11.2 Obtaining custom Time Stamps for Security Events . . . 444
9.11.3 Deployment for Security Events . . . 444

10 Service Instance Manifest 447

10.1 Service Interface Deployment . . . 447
10.1.1 SOME/IP Service Interface Deployment . . . 451
10.1.2 DDS Service Interface Deployment . . . 459
10.1.3 User Defined Service Interface . . . 462
10.2 Service Instance Deployment . . . 466
10.2.1 SOME/IP Service Instance Deployment . . . 473
10.2.1.1 Provided Service Instance . . . 474
10.2.1.2 Required Service Instance . . . 494
10.2.2 DDS Service Instance Deployment . . . 505
10.2.2.1 Provided DDS Service Instance . . . 507
10.2.2.2 Required DDS Service Instance . . . 509
10.2.2.3 DDS Service Instance to Machine mapping . . . 511
10.2.3 User Defined Service Instance Deployment . . . 512
10.3 EndToEndProtection . . . 514
10.4 Secure Communication . . . 522
10.4.1 Secure Communication over TLS . . . 526
10.4.2 Secure Communication over SecOC . . . 534
10.5 Raw Data Stream . . . 536
10.5.1 Raw Data Stream Deployment . . . 536
10.5.2 Raw Data Stream Mapping . . . 537

11 Signal-based communication 540

11.1 Overview . . . 540
11.2 Signal-based prerequisites . . . 543
11.3 Signal-based Deployment . . . 544
11.4 Signal-To-Service Mapping . . . 545
11.4.1 SignalBasedEvent Mapping . . . 547
11.4.2 SignalBasedField Mapping . . . 551
11.4.3 SignalBasedMethod Mapping . . . 555

11.5 Service discovery control . . . 557
11.5.1 Service control right after translation start . . . 559
11.5.2 Service control due to availability of related service instance 560
11.5.2.1 Signal to Service . . . 560
11.5.2.2 Service to Signal . . . 560
11.6 Translation behavior . . . 561
11.6.1 Translation from one source . . . 562
11.6.2 Translation from several sources . . . 563
11.7 Translation pass-through composition . . . 564
11.8 Expected features of Classic platform . . . 564
11.8.1 Processing order . . . 565
11.8.2 Reception data filter . . . 565
11.8.3 Reception of invalid signal . . . 565
11.8.4 Update Bit handling . . . 566
11.8.5 Transfer properties and transmission modes for Service to Signal . . . 567
11.8.6 Deadline monitoring . . . 567
11.8.7 Signal and IPdu Transmission . . . 567
11.8.8 IPdu multiplexing . . . 567
11.9 End-to-End considerations . . . 567
11.9.1 Safety . . . 568
11.9.1.1 Signal to Service . . . 571
11.9.1.2 Service to Signal . . . 571
11.9.2 Security . . . 572

12 Cross-FunctionalCluster interaction 574

12.1 ComCertificateToCryptoCertificateMapping . . . 576
12.2 ComKeyToCryptoKeySlotMapping . . . 577
12.3 ComSecOcToCryptoKeySlotMapping . . . 579
12.4 PersistencyDeploymentToCryptoKeySlotMapping . . . 581
12.5 PersistencyDeploymentElementToCryptoKeySlotMapping . . . 582
12.6 PersistencyDeploymentToDltLogChannelMapping . . . 583

13 REST 585

13.1 REST Design . . . 585
13.1.1 Overview . . . 585
13.1.2 REST Service Interface . . . 588
13.1.3 REST Resource . . . 588
13.1.4 REST Element . . . 593
13.2 REST Service Deployment . . . 599

14 Software Distribution 605

14.1 Overview . . . 605
14.2 Software Cluster . . . 606
14.2.1 Software Cluster General Modeling . . . 606
14.2.2 Relevance of Software Cluster for Diagnostics . . . 611
14.2.3 Sub Software Cluster . . . 616

14.2.4 Software Cluster Dependency . . . 617
14.2.5 References between Software Clusters . . . 622
14.3 Software Package . . . 624
14.4 Vehicle Package . . . 628
14.4.1 Overview . . . 628
14.4.2 VehicleRolloutStep . . . 632
14.4.3 UcmStep . . . 633
14.4.4 SoftwarePackageStep . . . 634
14.4.5 Examples for the Usage of SoftwarePackageStep . . . 637
14.4.5.1 Examples for the Usage of transfer and process . . . 637
14.4.5.2 Examples for the Usage of pre-activate and verify . . . 639

A Examples 644

A.1 Service Instance Deployment by Service Interface Mapping . . . 644
A.2 Service Instance Deployment by Service Interface Element Mapping . . . 646
A.3 Definition of Startup Configuration . . . 650
A.4 Service Instance Mapping . . . 653
A.5 Radar and Camera ServiceInterface example . . . 656
A.6 Definition of Persistent Data . . . 661
A.7 Definition of Persistent File . . . 663
A.8 Definition of Phm interaction . . . 665
A.8.1 Phm Application Design example . . . 665
A.8.2 Phm configuration example . . . 666
A.9 Scenarios to define a Vector . . . 669

B Custom Model Extension 670

B.1 Overview . . . 670
B.2 Custom Attribute Definition . . . 672
B.2.1 Custom Primitive Attribute Definition . . . 672
B.2.2 Custom Complex Attribute Definition . . . 674
B.3 Custom Foreign Reference Definition . . . 676
B.4 Custom Subclass Configuration . . . 679
B.5 Custom Constraints . . . 681
B.6 Definition of Reference from SdgClass to SdgClass . . . 682

C General Modeling 685

C.1 Reference to a DataPrototype in a PortInterface . . . 685
C.1.1 Reference to the inside of an ApplicationDataType . . . 685
C.1.2 Reference to the inside of a CppImplementationDataType 688
C.2 Reference to a AutosarDataPrototype in an Executable . . . 690
C.3 Reference to a PortPrototype in an Executable . . . 696
C.4 Modeling of a Method in an Executable . . . 705
C.5 Modeling of Mode-related InstanceRefs . . . 706
C.6 Modeling of Diagnostic-related InstanceRefs . . . 709
C.7 Modeling of REST-related InstanceRefs . . . 713
C.8 Modeling of PHM-related InstanceRefs . . . 714
C.9 Modeling of Time-related InstanceRefs . . . 718

C.10 Modeling of Persistency-related InstanceRefs . . . 719
C.11 Modeling of diagnostic-related InstanceRefs . . . 720
C.12 Modeling of SoftwareClusterDesign-related InstanceRefs . . . 726

D Mentioned Class Tables 729

E History of Constraints and Specification Items 800

E.1 Constraint and Specification Item History of this document according to AUTOSAR Release R17-03 (original version) . . . 800
E.1.1 Created Constraints in R17-03 . . . 800
E.1.2 Created Specification Items in R17-03 . . . 803
E.2 Constraint and Specification Item History of this document according to AUTOSAR Release R17-10 . . . 807
E.2.1 Added Traceables in R17-10 . . . 807
E.2.2 Changed Traceables in R17-10 . . . 811
E.2.3 Deleted Traceables in R17-10 . . . 812
E.2.4 Added Constraints in R17-10 . . . 813
E.2.5 Changed Constraints in R17-10 . . . 814
E.2.6 Deleted Constraints in R17-10 . . . 815
E.3 Constraint and Specification Item History of this document according to AUTOSAR Release R18-03 . . . 815
E.3.1 Added Traceables in R18-03 . . . 815
E.3.2 Changed Traceables in R18-03 . . . 819
E.3.3 Deleted Traceables in R18-03 . . . 820
E.3.4 Added Constraints in R18-03 . . . 821
E.3.5 Changed Constraints in R18-03 . . . 823
E.3.6 Deleted Constraints in R18-03 . . . 824
E.4 Constraint and Specification Item History of this document according to AUTOSAR Release R18-10 . . . 824
E.4.1 Added Traceables in R18-10 . . . 824
E.4.2 Changed Traceables in R18-10 . . . 826
E.4.3 Deleted Traceables in R18-10 . . . 828
E.4.4 Added Constraints in R18-10 . . . 830
E.4.5 Changed Constraints in R18-10 . . . 832
E.4.6 Deleted Constraints in R18-10 . . . 833
E.5 Constraint and Specification Item History of this document according to AUTOSAR Release R19-03 . . . 835
E.5.1 Added Traceables in R19-03 . . . 835
E.5.2 Changed Traceables in R19-03 . . . 837
E.5.3 Deleted Traceables in R19-03 . . . 838
E.5.4 Added Constraints in R19-03 . . . 838
E.5.5 Changed Constraints in R19-03 . . . 839
E.5.6 Deleted Constraints in R19-03 . . . 840
E.6 Constraint and Specification Item History of this document according to AUTOSAR Release R19-11 . . . 841
E.6.1 Added Traceables in R19-11 . . . 841
E.6.2 Changed Traceables in R19-11 . . . 844

E.6.3 Deleted Traceables in R19-11 . . . 845
E.6.4 Added Constraints in R19-11 . . . 845
E.6.5 Changed Constraints in R19-11 . . . 846
E.6.6 Deleted Constraints in R19-11 . . . 847
E.7 Constraint and Specification Item History of this document according to AUTOSAR Release R20-11 . . . 847
E.7.1 Added Traceables in R20-11 . . . 847
E.7.2 Changed Traceables in R20-11 . . . 850
E.7.3 Deleted Traceables in R20-11 . . . 853
E.7.4 Added Constraints in R20-11 . . . 855
E.7.5 Changed Constraints in R20-11 . . . 857
E.7.6 Deleted Constraints in R20-11 . . . 858
F Splitable Elements in the Scope of this Document 860
G Variation Points in the Scope of this Document 862

References

[1] Software Component Template
AUTOSAR_TPS_SoftwareComponentTemplate

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture

[3] Reference Model for Service Oriented Architecture 1.0
https://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

[4] Standardization Template
AUTOSAR_TPS_StandardizationTemplate

[5] Specification of RESTful communication
AUTOSAR_SWS_REST

[6] Generic Structure Template
AUTOSAR_TPS_GenericStructureTemplate

[7] SOME/IP Protocol Specification
AUTOSAR_PRS_SOMEIPProtocol

[8] Specification of Communication Management
AUTOSAR_SWS_CommunicationManagement

[9] Specification of Persistency
AUTOSAR_SWS_Persistency

[10] IEEE Standard for Information Technology- Standardized Application Environment Profile (AEP)-POSIX Realtime and Embedded Application Support
https://standards.ieee.org/findstds/standard/1003.13-2003.html

[11] Specification of Time Synchronization for Adaptive Platform
AUTOSAR_SWS_TimeSync

[12] Specification of Platform Health Management for Adaptive Platform
AUTOSAR_SWS_PlatformHealthManagement

[13] Specification of Cryptography for Adaptive Platform
AUTOSAR_SWS_Cryptography

[14] Specification of Identity and Access Management
AUTOSAR_SWS_IdentityAndAccessManagement

[15] Explanation of ara::com API
AUTOSAR_EXP_ARAComAPI

[16] Information technology – Universal Coded Character Set (UCS)
http://www.iso.org

[17] System Template
AUTOSAR_TPS_SystemTemplate

[18] Specification of Execution Management
AUTOSAR_SWS_ExecutionManagement

[19] Diagnostic Extract Template
AUTOSAR_TPS_DiagnosticExtractTemplate

[20] Specification of Diagnostics
AUTOSAR_SWS_Diagnostics

[21] Security Extract Template
AUTOSAR_TPS_SecurityExtractTemplate

[22] Road vehicles – Diagnostic communication over Internet Protocol (DoIP)
http://www.iso.org

[23] Log and Trace Protocol Specification
AUTOSAR_PRS_LogAndTraceProtocol

[24] SOME/IP Service Discovery Protocol Specification
AUTOSAR_PRS_SOMEIPServiceDiscoveryProtocol

[25] Data Distribution Service (DDS), Version 1.4
http://www.omg.org/spec/DDS/1.4

[26] RPC over DDS, Version 1.0
https://www.omg.org/spec/DDS-RPC/1.0

[27] DDS Consolidated XML Syntax, Version 1.0
https://www.omg.org/spec/DDS-XML

[28] Specification of SW-C End-to-End Communication Protection Library
AUTOSAR_SWS_E2ELibrary

[29] E2E Protocol Specification
AUTOSAR_PRS_E2EProtocol

[30] Specification of Communication
AUTOSAR_SWS_COM

[31] Specification of Secure Onboard Communication
AUTOSAR_SWS_SecureOnboardCommunication

[32] REST: Architectural Styles and the Design of Network-based Software Architectures

1 Introduction

This document contains the specification of the so-called the *Manifest* on the *AUTOSAR adaptive platform*. A description of the overall modeling approach can be found in section 1.1. A reference to the definition of the term *service* is given in section 1.2.

The term *Manifest* is used in this specification in the meaning of a formal specification of configuration content. Please find a more detailed description of the term and the implications for the *AUTOSAR adaptive platform* in section 2.

Please note that the content of the document (despite the name) extends to the description of design elements necessary to develop software for the *AUTOSAR adaptive platform*.

The design-related modeling mainly is focused on the development of application software on the *AUTOSAR adaptive platform* as well as the connection between application and diagnostics and is described in detail[1] in section 3 and section 4.

Section 5, in particular, describes the big picture of *AUTOSAR classic platform* and *AUTOSAR adaptive platform* communicating via service-oriented communication.

Section 7 describes the options for configuring a machine by means of a *manifest*.

Section 8 represents that counterpart to section 3 on deployment level, it describes the content of the so-called *execution manifest*.

Section 9 contains a string of sub-sections that explain the manifest content of platform module functionality.

Section 10 provides a detailed description of how service-oriented communication shall be configured on *manifest* level.

Section 11 explains how signal-based communication can be transformed into service-oriented communication and vice versa in order to participate in the communication between ECUs on the *AUTOSAR classic platform*.

Section 13 describes the modeling of communication with web services following the REST pattern

Section 14 describes the idea behind and the configuration of the concept of an uploadable software package.

[1] The description of the design elements may be moved to other model-related documents in the future. But for the time being, there is a coexistence of manifest-related and design-related model elements in this document.

1.1 Modeling Approach

The *AUTOSAR adaptive platform* has been introduced when the *AUTOSAR classic platform* was already a stable and well-established standard in the automotive domain.

And yet, the *AUTOSAR adaptive platform* is no successor of the *AUTOSAR classic platform*. Both platforms complement each other for specific use cases that can be better implemented by one or the other platform.

In this situation, two possible approaches for modeling on the *AUTOSAR adaptive platform* could have been taken:

- The *AUTOSAR adaptive platform* is based on different principles than the *AUTOSAR classic platform*, and hence the modeling approach could also **decouple from the canon of the AUTOSAR classic platform as much as possible** to advertise the fact that the two platforms have different purposes.

  Consequentially, even if specific model elements have clear counterparts in the respective other platform, use a different terminology to not confuse the users of both platforms.

- Despite the undeniable differences between the two platforms, there is still a significant number of striking similarities that strongly encourage the **usage of existing modeling concepts** from the *AUTOSAR classic platform*, especially from the specification of the AUTOSAR Software-Component Template [1], as much as possible.

  Consequentially, the conclusion is to use the identical meta-classes for similar purposes on both platforms. It will then be necessary to extend some of the affected meta-classes platform specific where applicable and add constraints that clarify the platform-specific usage of the mentioned extensions.

Without further ado, the modeling approach for the *AUTOSAR adaptive platform* follows the second alternative.

This means, for example, that a piece of application software on the *AUTOSAR adaptive platform* shall be represented by an `SwComponentType`. This includes the definition of `CompositionSwComponentType`s that in turn aggregate `SwComponentPrototype`s typed by e.g. (in case of the *AUTOSAR adaptive platform*) `AdaptiveApplicationSwComponentType`s.

The reuse of existing model-elements for the definition of the meta-model for the *AUTOSAR adaptive platform* has the side effect that the descriptions of existing model elements may contain references to technical details that only make sense on the *AUTOSAR classic platform*.

After all, the model elements were created when only the *AUTOSAR classic platform* existed.

These references shall be taken with a grain of salt. It is expected that readers can abstract from those details and extract the aspects of these model elements that create relevance for the description of the *AUTOSAR adaptive platform*.

1.2 The Term Service

It is essential to keep in mind that the term *service* is frequently used within this document in particular and the *AUTOSAR adaptive platform* in general.

This usage has its reasons despite the fact that the meaning of the term *service* on the *AUTOSAR adaptive platform* collides with other meanings used within AUTOSAR.

In summary, the following meaning of the term *service* exist in the scope of AUTOSAR:

- The Term *service* is used in the layered software architecture [2] to denote the highest layer of the AUTOSAR software architecture that interacts with the application. In this context, model elements like `ServiceSwComponentType`, `SwcServiceDependency`, `ServiceNeeds`, or `PortInterface.isService` have been created on the *AUTOSAR classic platform*.
- The term *service* is used to express that information is related or required in a workshop where a car is **serviced**. In this context, *service-only diagnostic trouble codes* (DTC) are defined.
- The term *service* is used to describe the handling of **diagnostic services**, e.g. UDS service *ReadDataByIdentifier*, for the communication between a diagnostic tester and a diagnostic stack on an (AUTOSAR) ECU.
- the term *service* is used in the meaning defined by the **service-oriented architecture** (SOA) [3]. This meaning has the strongest relation to the usage of the term *service* on the *AUTOSAR adaptive platform*.

1.3 Abbreviations

The following table contains a list of abbreviations used in the scope of this document along with the spelled-out meaning of each of the abbreviations.

Abbreviation	*Meaning*
AES	Advanced Encryption Standard
API	Application Programming Interface
ATP	AUTOSAR Template Profile
ARXML	AUTOSAR XML

▽

△

Abbreviation	*Meaning*
CAN	Controller Area Network
CRC	Cyclic Redundancy Check
CTM	Counter Mode
DDS	Data Distribution Service
DES	Data Encryption Standard
DHCP	Dynamic Host Control Protocol
DoIP	Diagnostics over IP
DM	Diagnostic Manager
DTC	Diagnostic Trouble Code
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECU	Electrical Control Unit
ECIES	Elliptic Curve Integrated Encryption Scheme
EDDSA	Edwards-Curve Digital Signature Algorithm
FQDN	Fully-Qualified Domain Name
GCM	Galios/Counter Mode
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transport Protocol
ICMP	Internet Control Message Protocol
ID	Identifier
IO	Input/Output
IP	Internet Protocol
ISO	International Standardization Organization
JSON	JavaScript Object Notation
LAN	Local Area Network
MAC	Media Access Control
MAC	Message Authentication Code
MD	Message Digest
MTU	Maximum Transmission Unit
NM	Network Management
NV	Non-Volatile
OEM	Original Equipment Manufacturer
OS	Operating System
PDU	Protocol Data Unit

▽

△

Abbreviation	*Meaning*
PHM	Platform Health Management
PKCS	Public Key Cryptography Standards
POSIX	Portable Operating System Interface
PSK	Pre-Shared Key
RAM	Random Access Memory
REST	Representational State Transfer
ROM	Read-Only Memory
RSA	Cryptographic approach according to Rivest, Shamir, and Adleman
SD	Service Discovery
SDG	Special Data Group
SHA	Secure Hash Algorithm
SOME/IP	Scalable service-Oriented MiddlewarE over IP
SWC	Software Component
TCP	Transport Control Protocol
TLS	Transport Layer Security
TLV	Tag Length Value
TTL	Time to Live
UDS	Unified Diagnostic Services
UDP	User datagram Protocol
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
VFB	Virtual Functional Bus
VLAN	Virtual Local Area Network
VSA	Variable Size Array
XML	Extensible Markup Language
XSD	XML Schema Definition

Table 1.1: Abbreviations used in the scope of this Document

1.4 Document Conventions

Technical terms are typeset in mono spaced font, e.g. `PortPrototype`. As a general rule, plural forms of technical terms are created by adding "s" to the singular form, e.g.

PortPrototypes. By this means the document resembles terminology used in the AUTOSAR XML Schema.

This document contains constraints in textual form that are distinguished from the rest of the text by a unique numerical constraint ID, a headline, and the actual constraint text starting after the ⌈ character and terminated by the ⌋ character.

The purpose of these constraints is to literally constrain the interpretation of the AUTOSAR meta-model such that it is possible to detect violations of the standardized behavior implemented in an instance of the meta-model (i.e. on M1 level).

Makers of AUTOSAR tools are encouraged to add the numerical ID of a constraint that corresponds to an M1 modeling issue as part of the diagnostic message issued by the tool.

The attributes of the classes introduced in this document are listed in form of class tables. They have the form shown in the example of the top-level element AUTOSAR:

Please note that constraints are not supposed to be enforceable at any given time in an AUTOSAR workflow. During the development of a model, constraints may legitimately be violated because an incomplete model will obviously show inconsistencies.

However, at specific points in the workflow, constraints shall be enforced as a safeguard against misconfiguration.

The points in the workflow where constraints shall be enforced, sometimes also known as the "binding time" of the constraint, are different for each model category, e.g. on the classic platform, the constraints defined for software-components are typically enforced prior to the generation of the RTE while the constraints against the definition of an Ecu extract shall be applied when the Ecu configuration for the Com stack is created.

For each document, possible binding times of constraints are defined and the binding times are typically mentioned in the constraint themselves to give a proper orientation for implementers of AUTOSAR authoring tools.

Class	**AUTOSAR**			
Package	M2::AUTOSARTemplates::AutosarTopLevelStructure			
Note	Root element of an AUTOSAR description, also the root element in corresponding XML documents. **Tags:**xml.globalElement=true			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data of an Autosar file. **Tags:**xml.sequenceOffset=10

▽

△

Class	**AUTOSAR**			
arPackage	ARPackage	*	aggr	This is the top level package in an AUTOSAR model. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=arPackage.shortName, arPackage.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
fileInfoComment	FileInfoComment	0..1	aggr	This represents a possibility to provide a structured comment in an AUTOSAR file. **Stereotypes:** atpStructuredComment **Tags:** xml.roleElement=true xml.sequenceOffset=-10 xml.typeElement=false
introduction	DocumentationBlock	0..1	aggr	This represents an introduction on the Autosar file. It is intended for example to rpresent disclaimers and legal notes. **Tags:**xml.sequenceOffset=20

Table 1.2: AUTOSAR

The first rows in the table have the following meaning:

Class: The name of the class as defined in the UML model.

Package: The UML package the class is defined in. This is only listed to help locating the class in the overall meta model.

Note: The comment the modeler gave for the class (class note). Stereotypes and UML tags of the class are also denoted here.

Base Classes: If applicable, the list of direct base classes.

The headers in the table have the following meaning:

Attribute: The name of an attribute of the class. Note that AUTOSAR does not distinguish between class attributes and owned association ends.

Type: The type of an attribute of the class.

Mul.: The assigned multiplicity of the attribute, i.e. how many instances of the given data type are associated with the attribute.

Kind: Specifies, whether the attribute is aggregated in the class (`aggr` aggregation), an UML attribute in the class (`attr` primitive attribute), or just referenced by it (`ref` reference). Instance references are also indicated (`iref` instance reference) in this field.

Note: The comment the modeler gave for the class attribute (role note). Stereotypes and UML tags of the class are also denoted here.

Please note that the chapters that start with a letter instead of a numerical value represent the appendix of the document. The purpose of the appendix is to support the explanation of certain aspects of the document and does not represent binding con-

ventions of the standard. The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see Standardization Template, chapter Support for Traceability ([4]).

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see Standardization Template, chapter Support for Traceability ([4]).

1.5 Requirements Tracing

Requirements against this document are exclusively stated in the corresponding requirements document.

The following table 1.3 references the requirements specified in the corresponding requirements document and provides information about individual specification items that fulfill a given requirement.

Requirement	Description	Satisfied by
[RS_MANI_00002]	Declaration of provided and required services in an application	[TPS_MANI_01039] [TPS_MANI_01040] [TPS_MANI_01053] [TPS_MANI_01057] [TPS_MANI_03210] [TPS_MANI_03211] [TPS_MANI_03212]
[RS_MANI_00003]	Specification of service interfaces	[TPS_MANI_01001] [TPS_MANI_01004] [TPS_MANI_01005] [TPS_MANI_01006] [TPS_MANI_01007] [TPS_MANI_01033] [TPS_MANI_01034] [TPS_MANI_01035] [TPS_MANI_01064] [TPS_MANI_03118] [TPS_MANI_03119] [TPS_MANI_03223]
[RS_MANI_00004]	Support of application design	[TPS_MANI_01010]
[RS_MANI_00005]	Configuration of diagnostic capabilities of an application	[TPS_MANI_01037] [TPS_MANI_01048] [TPS_MANI_01049] [TPS_MANI_01050] [TPS_MANI_01060] [TPS_MANI_01259] [TPS_MANI_01260] [TPS_MANI_01261] [TPS_MANI_01262] [TPS_MANI_01263] [TPS_MANI_01326] [TPS_MANI_01336]
[RS_MANI_00006]	Support of application deployment	[TPS_MANI_01011] [TPS_MANI_01337]
[RS_MANI_00007]	Configuration of application startup behavior	[TPS_MANI_01012] [TPS_MANI_01013] [TPS_MANI_01014] [TPS_MANI_01017] [TPS_MANI_01041] [TPS_MANI_01046] [TPS_MANI_01061] [TPS_MANI_01188] [TPS_MANI_01209] [TPS_MANI_01277] [TPS_MANI_01278]
[RS_MANI_00008]	Service interface deployment to a transport layer mechanism	[TPS_MANI_01136] [TPS_MANI_01137] [TPS_MANI_01210] [TPS_MANI_03036] [TPS_MANI_03037] [TPS_MANI_03038] [TPS_MANI_03039] [TPS_MANI_03070] [TPS_MANI_03071] [TPS_MANI_03072] [TPS_MANI_03073] [TPS_MANI_03074] [TPS_MANI_03075] [TPS_MANI_03101] ▽

▽

△

Requirement	Description	Satisfied by
		△ [TPS_MANI_03103] [TPS_MANI_03104] [TPS_MANI_03105] [TPS_MANI_03106] [TPS_MANI_03107] [TPS_MANI_03108] [TPS_MANI_03116] [TPS_MANI_03117] [TPS_MANI_03217] [TPS_MANI_03235]
[RS_MANI_00009]	Service instance configuration on the network-level	[TPS_MANI_01316] [TPS_MANI_01317] [TPS_MANI_03001] [TPS_MANI_03002] [TPS_MANI_03003] [TPS_MANI_03004] [TPS_MANI_03007] [TPS_MANI_03008] [TPS_MANI_03009] [TPS_MANI_03010] [TPS_MANI_03022] [TPS_MANI_03023] [TPS_MANI_03024] [TPS_MANI_03049] [TPS_MANI_03061] [TPS_MANI_03237] [TPS_MANI_03554] [TPS_MANI_03555]
[RS_MANI_00011]	Instantiation of provided and required services in an application	[TPS_MANI_03000]
[RS_MANI_00014]	User defined transport layer mechanisms	[TPS_MANI_01165] [TPS_MANI_03032] [TPS_MANI_03045] [TPS_MANI_03046] [TPS_MANI_03047] [TPS_MANI_03048] [TPS_MANI_03102]
[RS_MANI_00015]	Definition of the nature of a manifest	[TPS_MANI_01000] [TPS_MANI_01019] [TPS_MANI_01020] [TPS_MANI_01021]
[RS_MANI_00016]	Usage of data types specifically on the AUTOSAR adaptive platform	[TPS_MANI_01016] [TPS_MANI_01027] [TPS_MANI_01047] [TPS_MANI_01100]
[RS_MANI_00017]	Specification of the mapping of Service Interfaces	[TPS_MANI_01002] [TPS_MANI_01003] [TPS_MANI_01022] [TPS_MANI_01024] [TPS_MANI_01025] [TPS_MANI_01026] [TPS_MANI_01032]
[RS_MANI_00018]	Network connections of the machine	[TPS_MANI_03052] [TPS_MANI_03053]
[RS_MANI_00019]	Service discovery message exchange configuration	[TPS_MANI_03064]
[RS_MANI_00020]	Hardware resources of the machine	[TPS_MANI_03035]
[RS_MANI_00021]	Description of machine states	[TPS_MANI_03035]
[RS_MANI_00022]	Adaptive Platform configuration	[TPS_MANI_01208] [TPS_MANI_03035]
[RS_MANI_00023]	Adaptive Module configuration	[TPS_MANI_01208] [TPS_MANI_01226] [TPS_MANI_01227] [TPS_MANI_01271] [TPS_MANI_01279] [TPS_MANI_03035] [TPS_MANI_03056] [TPS_MANI_03096] [TPS_MANI_03098] [TPS_MANI_03162] [TPS_MANI_03163] [TPS_MANI_03164] [TPS_MANI_03165] [TPS_MANI_03166] [TPS_MANI_03167] [TPS_MANI_03218] [TPS_MANI_03219] [TPS_MANI_03220] [TPS_MANI_03221] [TPS_MANI_03222] [TPS_MANI_03226] [TPS_MANI_03260] [TPS_MANI_03261] [TPS_MANI_03262] [TPS_MANI_03263] [TPS_MANI_03264] [TPS_MANI_03265] [TPS_MANI_03266] [TPS_MANI_03267] [TPS_MANI_03274] [TPS_MANI_03502] [TPS_MANI_03503] [TPS_MANI_03505] [TPS_MANI_03506] [TPS_MANI_03508] [TPS_MANI_03509] [TPS_MANI_03510] [TPS_MANI_03511] [TPS_MANI_03512] [TPS_MANI_03513] [TPS_MANI_03514] [TPS_MANI_03515] ▽

▽

△

Requirement	Description	Satisfied by
		△ [TPS_MANI_03516] [TPS_MANI_03517] [TPS_MANI_03544] [TPS_MANI_03545] [TPS_MANI_03546] [TPS_MANI_03552] [TPS_MANI_03553] [TPS_MANI_03573] [TPS_MANI_03574] [TPS_MANI_03575] [TPS_MANI_03576] [TPS_MANI_03625] [TPS_MANI_03626]
[RS_MANI_00024]	SOME/IP transport layer mechanisms	[TPS_MANI_01136] [TPS_MANI_01137] [TPS_MANI_03002] [TPS_MANI_03003] [TPS_MANI_03004] [TPS_MANI_03007] [TPS_MANI_03008] [TPS_MANI_03009] [TPS_MANI_03010] [TPS_MANI_03011] [TPS_MANI_03012] [TPS_MANI_03013] [TPS_MANI_03014] [TPS_MANI_03015] [TPS_MANI_03016] [TPS_MANI_03017] [TPS_MANI_03018] [TPS_MANI_03020] [TPS_MANI_03021] [TPS_MANI_03022] [TPS_MANI_03023] [TPS_MANI_03024] [TPS_MANI_03025] [TPS_MANI_03026] [TPS_MANI_03027] [TPS_MANI_03028] [TPS_MANI_03029] [TPS_MANI_03030] [TPS_MANI_03031] [TPS_MANI_03040] [TPS_MANI_03041] [TPS_MANI_03042] [TPS_MANI_03043] [TPS_MANI_03044] [TPS_MANI_03049] [TPS_MANI_03050] [TPS_MANI_03051] [TPS_MANI_03057] [TPS_MANI_03059] [TPS_MANI_03061] [TPS_MANI_03067] [TPS_MANI_03068] [TPS_MANI_03069] [TPS_MANI_03070] [TPS_MANI_03071] [TPS_MANI_03072] [TPS_MANI_03073] [TPS_MANI_03074] [TPS_MANI_03075] [TPS_MANI_03116] [TPS_MANI_03154] [TPS_MANI_03155] [TPS_MANI_03156] [TPS_MANI_03157] [TPS_MANI_03158] [TPS_MANI_03159] [TPS_MANI_03168] [TPS_MANI_03217] [TPS_MANI_03227] [TPS_MANI_03230] [TPS_MANI_03231] [TPS_MANI_03235] [TPS_MANI_03237] [TPS_MANI_03554] [TPS_MANI_03555]
[RS_MANI_00025]	Definition and configuration of serialization	[TPS_MANI_01210] [TPS_MANI_03101] [TPS_MANI_03102] [TPS_MANI_03103] [TPS_MANI_03104] [TPS_MANI_03105] [TPS_MANI_03106] [TPS_MANI_03107] [TPS_MANI_03108] [TPS_MANI_03117]
[RS_MANI_00026]	Software Component System Design	[TPS_MANI_01054] [TPS_MANI_01191] [TPS_MANI_01192] [TPS_MANI_01198] [TPS_MANI_03110] [TPS_MANI_03111] [TPS_MANI_03112] [TPS_MANI_03113] [TPS_MANI_03114] [TPS_MANI_03115]
[RS_MANI_00027]	Support for access to persistent data	[TPS_MANI_01065] [TPS_MANI_01067] [TPS_MANI_01068] [TPS_MANI_01073] [TPS_MANI_01078] [TPS_MANI_01079] [TPS_MANI_01080] [TPS_MANI_01081] [TPS_MANI_01135] [TPS_MANI_01138] [TPS_MANI_01139] [TPS_MANI_01140] [TPS_MANI_01142] [TPS_MANI_01144] [TPS_MANI_01146] [TPS_MANI_01147] ▽

▽

△

Requirement	Description	Satisfied by
		△ [TPS_MANI_01148] [TPS_MANI_01149] [TPS_MANI_01150] [TPS_MANI_01155] [TPS_MANI_01156] [TPS_MANI_01157] [TPS_MANI_01159] [TPS_MANI_01160] [TPS_MANI_01179] [TPS_MANI_01180] [TPS_MANI_01182] [TPS_MANI_01187] [TPS_MANI_01194] [TPS_MANI_01196] [TPS_MANI_01197] [TPS_MANI_01204] [TPS_MANI_01205] [TPS_MANI_01206] [TPS_MANI_01207] [TPS_MANI_01313] [TPS_MANI_01314] [TPS_MANI_01315] [TPS_MANI_01319] [TPS_MANI_01320] [TPS_MANI_01321] [TPS_MANI_01322] [TPS_MANI_01323]
[RS_MANI_00028]	Configuration of Safety protection	[TPS_MANI_01324] [TPS_MANI_01325] [TPS_MANI_01327] [TPS_MANI_03127] [TPS_MANI_03128] [TPS_MANI_03129] [TPS_MANI_03130] [TPS_MANI_03131] [TPS_MANI_03132] [TPS_MANI_03228] [TPS_MANI_03229] [TPS_MANI_03252]
[RS_MANI_00029]	Mapping description between Signal-based communication and Service-Oriented communication	[TPS_MANI_03124] [TPS_MANI_03125] [TPS_MANI_03126] [TPS_MANI_03627] [TPS_MANI_03629]
[RS_MANI_00030]	Definition of optional elements in composite data structures	[TPS_MANI_01097] [TPS_MANI_01184] [TPS_MANI_01185] [TPS_MANI_01186] [TPS_MANI_01270] [TPS_MANI_01333]
[RS_MANI_00031]	Interaction with Crypto Software	[TPS_MANI_03253] [TPS_MANI_03254] [TPS_MANI_03255] [TPS_MANI_03256] [TPS_MANI_03257] [TPS_MANI_03258] [TPS_MANI_03259]
[RS_MANI_00032]	Support for platform health management	[TPS_MANI_03500] [TPS_MANI_03502] [TPS_MANI_03503] [TPS_MANI_03505] [TPS_MANI_03506] [TPS_MANI_03508] [TPS_MANI_03509] [TPS_MANI_03510] [TPS_MANI_03511] [TPS_MANI_03512] [TPS_MANI_03513] [TPS_MANI_03514] [TPS_MANI_03515] [TPS_MANI_03516] [TPS_MANI_03517] [TPS_MANI_03534] [TPS_MANI_03544] [TPS_MANI_03545] [TPS_MANI_03546] [TPS_MANI_03552] [TPS_MANI_03553] [TPS_MANI_03573] [TPS_MANI_03574] [TPS_MANI_03575] [TPS_MANI_03576] [TPS_MANI_03625] [TPS_MANI_03626]
[RS_MANI_00033]	Interaction with web services based on the REST pattern	[TPS_MANI_01103] [TPS_MANI_01105] [TPS_MANI_01120] [TPS_MANI_01121] [TPS_MANI_01122] [TPS_MANI_01123] [TPS_MANI_01124] [TPS_MANI_01125] [TPS_MANI_01126] [TPS_MANI_01127] [TPS_MANI_01128] [TPS_MANI_01129] [TPS_MANI_01130] [TPS_MANI_01131] [TPS_MANI_01178]
[RS_MANI_00034]	Specification of intents	[TPS_MANI_01106] [TPS_MANI_01107] [TPS_MANI_01108] [TPS_MANI_03209]

▽

△

Requirement	Description	Satisfied by
[RS_MANI_00035]	Definition of an uploadable software package	[TPS_MANI_01109] [TPS_MANI_01110] [TPS_MANI_01111] [TPS_MANI_01112] [TPS_MANI_01113] [TPS_MANI_01114] [TPS_MANI_01115] [TPS_MANI_01116] [TPS_MANI_01117] [TPS_MANI_01118] [TPS_MANI_01119] [TPS_MANI_01161] [TPS_MANI_01164] [TPS_MANI_01189] [TPS_MANI_01202] [TPS_MANI_01211] [TPS_MANI_01213] [TPS_MANI_01214] [TPS_MANI_01215] [TPS_MANI_01216] [TPS_MANI_01217] [TPS_MANI_01218] [TPS_MANI_01219] [TPS_MANI_01220] [TPS_MANI_01221] [TPS_MANI_01222] [TPS_MANI_01223] [TPS_MANI_01225] [TPS_MANI_01335]
[RS_MANI_00036]	Configuration of security protection	[TPS_MANI_03133] [TPS_MANI_03134] [TPS_MANI_03137] [TPS_MANI_03138] [TPS_MANI_03139] [TPS_MANI_03140] [TPS_MANI_03199] [TPS_MANI_03200] [TPS_MANI_03203] [TPS_MANI_03204] [TPS_MANI_03205] [TPS_MANI_03206] [TPS_MANI_03208] [TPS_MANI_03213] [TPS_MANI_03214] [TPS_MANI_03216] [TPS_MANI_03232] [TPS_MANI_03233]
[RS_MANI_00037]	Configuration of logging and tracing	[TPS_MANI_01272] [TPS_MANI_03160]
[RS_MANI_00038]	DDS transport layer mechanisms	[TPS_MANI_03525] [TPS_MANI_03526] [TPS_MANI_03527] [TPS_MANI_03528] [TPS_MANI_03529] [TPS_MANI_03530] [TPS_MANI_03531] [TPS_MANI_03532] [TPS_MANI_03533] [TPS_MANI_03556] [TPS_MANI_03557] [TPS_MANI_03558] [TPS_MANI_03561] [TPS_MANI_03562] [TPS_MANI_03567] [TPS_MANI_03568] [TPS_MANI_03622]
[RS_MANI_00039]	Usage of implementation specific data types	[TPS_MANI_01166] [TPS_MANI_01167] [TPS_MANI_01168] [TPS_MANI_01169] [TPS_MANI_01171] [TPS_MANI_01172] [TPS_MANI_01173] [TPS_MANI_01174] [TPS_MANI_01175] [TPS_MANI_01176] [TPS_MANI_01177] [TPS_MANI_01201] [TPS_MANI_01212] [TPS_MANI_03169] [TPS_MANI_03170] [TPS_MANI_03171] [TPS_MANI_03172] [TPS_MANI_03173] [TPS_MANI_03174] [TPS_MANI_03175] [TPS_MANI_03176] [TPS_MANI_03177] [TPS_MANI_03178] [TPS_MANI_03179] [TPS_MANI_03180] [TPS_MANI_03181] [TPS_MANI_03183] [TPS_MANI_03184] [TPS_MANI_03185] [TPS_MANI_03186] [TPS_MANI_03187] [TPS_MANI_03188] [TPS_MANI_03189] [TPS_MANI_03190] [TPS_MANI_03191] [TPS_MANI_03192] [TPS_MANI_03193] [TPS_MANI_03196] [TPS_MANI_03197] [TPS_MANI_03198] [TPS_MANI_03201] [TPS_MANI_03202]

▽

△

Requirement	Description	Satisfied by
[RS_MANI_00040]	Support for access to synchronized time	[TPS_MANI_03535] [TPS_MANI_03536] [TPS_MANI_03537] [TPS_MANI_03539] [TPS_MANI_03541] [TPS_MANI_03542] [TPS_MANI_03543] [TPS_MANI_03547] [TPS_MANI_03548] [TPS_MANI_03549] [TPS_MANI_03551] [TPS_MANI_03632]
[RS_MANI_00041]	Configuration of function groups	[TPS_MANI_01330] [TPS_MANI_03145] [TPS_MANI_03152] [TPS_MANI_03194] [TPS_MANI_03195]
[RS_MANI_00050]	Support of Deterministic Client	[TPS_MANI_01199] [TPS_MANI_01200] [TPS_MANI_01203]
[RS_MANI_00060]	Support of Identity and Access Management	[TPS_MANI_01231] [TPS_MANI_01232] [TPS_MANI_01233] [TPS_MANI_01234] [TPS_MANI_01235] [TPS_MANI_01237] [TPS_MANI_01238] [TPS_MANI_01239] [TPS_MANI_01240] [TPS_MANI_01241]
[RS_MANI_00061]	Support of Diagnostic Interfaces	[TPS_MANI_01048] [TPS_MANI_01049] [TPS_MANI_01050] [TPS_MANI_01242] [TPS_MANI_01243] [TPS_MANI_01245] [TPS_MANI_01246] [TPS_MANI_01247] [TPS_MANI_01248] [TPS_MANI_01249] [TPS_MANI_01250] [TPS_MANI_01251] [TPS_MANI_01252] [TPS_MANI_01253] [TPS_MANI_01254] [TPS_MANI_01255] [TPS_MANI_01256] [TPS_MANI_01257] [TPS_MANI_01258] [TPS_MANI_01259] [TPS_MANI_01260] [TPS_MANI_01261] [TPS_MANI_01262] [TPS_MANI_01263] [TPS_MANI_01265] [TPS_MANI_01311] [TPS_MANI_01312] [TPS_MANI_01326] [TPS_MANI_01332] [TPS_MANI_01336]
[RS_MANI_00062]	Support for Partial Networking	[TPS_MANI_03224] [TPS_MANI_03225]
[RS_MANI_00063]	The Manifest specification shall support the translation between signal-based and service-oriented communication	[TPS_MANI_03577] [TPS_MANI_03578] [TPS_MANI_03579] [TPS_MANI_03580] [TPS_MANI_03581] [TPS_MANI_03582] [TPS_MANI_03583] [TPS_MANI_03584] [TPS_MANI_03585] [TPS_MANI_03586] [TPS_MANI_03587] [TPS_MANI_03588] [TPS_MANI_03589] [TPS_MANI_03590] [TPS_MANI_03591] [TPS_MANI_03592] [TPS_MANI_03593] [TPS_MANI_03594] [TPS_MANI_03595] [TPS_MANI_03596] [TPS_MANI_03597] [TPS_MANI_03598] [TPS_MANI_03599] [TPS_MANI_03600] [TPS_MANI_03601] [TPS_MANI_03602] [TPS_MANI_03603] [TPS_MANI_03604] [TPS_MANI_03605] [TPS_MANI_03606] [TPS_MANI_03607] [TPS_MANI_03608] [TPS_MANI_03609] [TPS_MANI_03610] [TPS_MANI_03611] [TPS_MANI_03612] [TPS_MANI_03614] [TPS_MANI_03615] [TPS_MANI_03620] [TPS_MANI_03621]
[RS_MANI_00064]	Service contract version for a service interface	[TPS_MANI_03616]
[RS_MANI_00065]	Service contract versioning for all Transport Deployment Protocols	[TPS_MANI_03617]
[RS_MANI_00066]	Service Versioning Blacklist	[TPS_MANI_03618]

▽

△

Requirement	Description	Satisfied by
[RS_MANI_00067]	Raw data stream deployment	[TPS_MANI_01285] [TPS_MANI_01287]

Table 1.3: RequirementsTracing

1.6 Known Limitations

The AUTOSAR SWS REST [5] defines a low-level API for REST-based communication. The content of section 13, on the other hand, applies for the configuration of a not-yet standardized API on top of the ara::rest API.

2 Big Picture of Manifest Definition

2.1 Design vs. Deployment

2.1.1 Overview

Despite the name, this document contains the description of model elements that are clearly bound to a *design* workflow **and** model elements that have a strong relation to the *deployment* aspect.

Model elements discussed in this document are either related to *design* or *deployment*, there is no overlap between the two groups.

Model elements that are related to *deployment* will be used in models that are uploaded to a target platform, see [TPS_MANI_01000]. These model elements are mainly described in sections of this document where the term “Manifest” is part of the section title.

In the absence of a more precise definition, model elements related to *design* can be identified by not being related to *deployment*.

2.1.2 Relation between Design and Deployment Models

Please note that in many cases the part of the meta-model related to *deployment* reflects a similar modeling in the *design* domain, e.g. the definition of E2E profile parameters.

There is currently no clearly defined preference about how the relation between *design* and *deployment* may impact a concrete development project. The following scenarios for the example of *E2E properties* might occur:

- An OEM delivers the description of `AdaptivePlatformServiceInstance`s including the definition of *E2E properties*.

  It is safe to assume that subsequent processing of the model shall take the *E2E properties* as granted and develop the software with respect to the given properties.

- Software exists that has defined *E2E properties* by means of `ComSpec`s. For various reasons, it may happen that the software cannot be updated and therefore takes the “lead” in terms of the definition of *E2E properties*.

  The definition of `AdaptivePlatformServiceInstance`s may then have to respect the existing modeling on the software side.

- It could also happen that existing definitions can be **partly** overwritten by engineers who **really** know what they are doing.

2.1.3 Structure of the document

The structure of the document maps to the division between *design* and *deployment* such that the *design* aspect is mostly described in sections 3, 4, 5, 6, most of 11, and 13.1.

In contrast, chapters 7, 8, 9, 10, 11.3, 12, 13.2, and 14 focus on *deployment*-related content.

2.2 About Manifest

This chapter shall clarify the definition of the term `Manifest` in the context of the *AUTOSAR adaptive platform*.

[TPS_MANI_01000]{DRAFT} **Definition of the term `Manifest`** ⌈A `Manifest` represents a piece of AUTOSAR model description that is created to support the configuration of an *AUTOSAR adaptive platform* product and which is uploaded to the *AUTOSAR adaptive platform* product, potentially in combination with other artifacts (like binary files) that contain executable code to which the `Manifest` applies.⌋*(RS_MANI_00015)*

It is important to stress the fact that the usage of a `Manifest` is indeed strictly limited to the *AUTOSAR adaptive platform* and that there is no use case to port the concept to the *AUTOSAR classic platform*.

2.3 Serialization Format

One aspect that the definition of a `Manifest` has in common with other AUTOSAR model content is the standardized serialization format.

[TPS_MANI_01020]{DRAFT} **Serialization format of the `Manifest` in AUTOSAR** ⌈The standardized serialization format of `Manifest` content in AUTOSAR is ARXML.

Consequently, `Manifest` model content can be validated against the AUTOSAR XML Schema.⌋*(RS_MANI_00015)*

An important consequence of [TPS_MANI_01020] is that there is no limitation to just one “manifest file” a.k.a. “the manifest”.

Content may be distributed among several physical files according to the rules given in the specification of the AUTOSAR Generic Structure Template [6].

[TPS_MANI_01021]{DRAFT} **Serialization format of `Manifest` content on a machine** ⌈The serialization format used to actually upload a manifest on a machine may be freely chosen by a platform supplier.

However, the content and semantics of the original ARXML `Manifest` needs to be **fully preserved**.⌋*(RS_MANI_00015)*

It can be expected that in many cases the best option for the upload of the `Manifest` will still be ARXML because a custom format obviously has to support the full complexity of the `Manifest` meta-model.

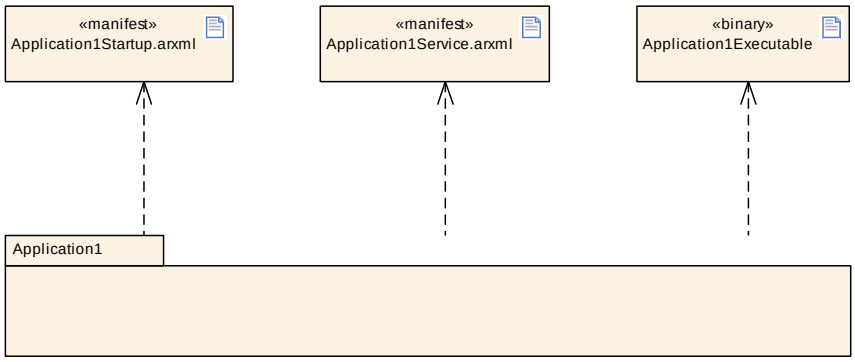


Figure 2.1: Example usage of several manifest files within one software delivery

Please note that the meta-model foresees the existence of references from manifest-related meta-classes to design-related meta-classes.

These references are created for the sake of clarity but it is not mandatory that the content of the reference actually needs to be resolvable.

In terms of the AUTOSAR modeling approach, this translates to a decoration of these references with the stereotype ≪`atpUriDef`≫. More information can be found in [6].

If the referenced meta-classes contain information that is relevant for the manifest level then this information is replicated on the manifest level (such that the manifest-level model does not have to rely on the availability of design-level information).

2.4 Scope

As mentioned before, the usage of a `Manifest` is limited to the *AUTOSAR adaptive platform*. This does not mean, however, that all ARXML produced in a development project that targets the *AUTOSAR adaptive platform* is automatically considered a `Manifest`.

In fact, the *AUTOSAR adaptive platform* is usually not exclusively used in a vehicle project.

A typical vehicle will most likely be also equipped with a number of ECUs developed on the *AUTOSAR classic platform* and the system design for the entire vehicle will therefore have to cover both ECUs built on top of the *AUTOSAR classic platform* and those created on top of the *AUTOSAR adaptive platform*.

[TPS_MANI_01019]{DRAFT} **`Manifest` content may apply to different aspects of the *AUTOSAR adaptive platform*** ⌈`Manifest` content can apply to different aspects

of the model. At the moment, Manifest content can roughly be divided into three focus areas:

- Application-related Manifest content describes all aspects of the deployment of an application, including - but not limited to - the startup configuration and the configuration of service-oriented communication endpoints on application level.
- Machine-related Manifest content describes the deployment of just a machine, i.e. without any application (including platform modules) running on the machine.
- Service instance-related Manifest describes how service-oriented communication on transport layer level is bound to endpoints in the application and (in some cases) platform software.

⌋(*RS_MANI_00015*)

2.5 Manifests described in this Document

In principle, the term Manifest could be defined such that there is conceptually just one “manifest” and every deployment aspect would be handled in this context.

This does not seem appropriate because it became apparent that manifest-related model-elements exist that are relevant in entirely different phases of a typical development project.

This aspect is taken as the main motivation to subdivide the definition of the term Manifest in three different partitions:

Execution Manifest This kind of Manifest is used to specify the deployment-related information of applications running on the *AUTOSAR adaptive platform*.

An Execution Manifest is bundled with the actual executable code in order to support the integration of the executable code onto the machine.

Please find more information regarding this topic in section 8.

Service Instance Manifest This kind of Manifest is used to specify how service-oriented communication is configured in terms of the requirements of the underlying transport protocols.

A Service Instance Manifest is bundled with the actual executable code that implements the respective usage of service-oriented communication.

Please find more information regarding this topic in section 10.

Machine Manifest This kind of Manifest is supposed to describe deployment-related content that applies to the configuration of just the underlying machine (i.e. without any applications running on the machine) that runs an *AUTOSAR adaptive platform*.

A `Machine Manifest` is bundled with the software taken to establish an instance of the *AUTOSAR adaptive platform*.

Please find more information regarding this topic in sections 7 and 9.

Software Distribution This kind of `Manifest` described the packaging and logistics aspects of software on the *AUTOSAR adaptive platform*.

Please find more information regarding this topic in section 14.

The temporal division between the definition (and usage) of different kinds of `Manifest` leads to the conclusion that in most cases different physical files will be used to store the content of the three kinds of `Manifest`.

However, as with all kinds of ARXML content, this is not a binding rule.

3 Application Design

3.1 Overview

This chapter describes all design-related modeling that applies to the creation of application software on the *AUTOSAR adaptive platform*.

This also extends to extensions of existing modeling used on the *AUTOSAR classic platform*, e.g. the introduction of new values of the attribute `category`.

In particular, this section of the document focuses on the following aspects:

- Definition of a dedicated subclass of `SwComponentType` for the *AUTOSAR adaptive platform* (section 3.2)
- Definition of data types specifically for the *AUTOSAR adaptive platform* (section 3.3)
- Service interface as the pivotal element for service-oriented communication (section 3.4)
- Service interface mapping as a mediator between internal and external communication (section 3.5)
- Service interface **element** mapping as a mediator between internal and external communication (section 3.6)
- Persistency interface as the basis for interacting with persistent data storage (section 3.8)
- Aspects of the fine-grained configuration of interaction with the “outside world” from the perspective of the inside of a software-component (section 3.15)
- `Executable` as the smallest executable unit (section 3.16)
- Configuration of transformation properties (section 3.18)

3.2 Software Component

In principle, it would be possible to directly take over the definition of e.g. `ApplicationSwComponentType` for the usage on the *AUTOSAR adaptive platform*.

However, this would complicate the formulation of constraints regarding the existence of model elements (for example: data types, as explained in section 3.3) that are exclusive to the *AUTOSAR adaptive platform*.

Therefore, the `AdaptiveApplicationSwComponentType` is defined as a representation of software-components on the *AUTOSAR adaptive platform*.

The Existence of the AdaptiveApplicationSwComponentType allows for a convenient way (see [constr_1492]) to lock out most kinds of software-component defined for the *AUTOSAR classic platform* from the usage on the *AUTOSAR adaptive platform*.

The clarification of the opposite direction (i.e. an erroneous use of an AdaptiveApplicationSwComponentType) is less obvious.

In other words, it may be possible to use an AdaptiveApplicationSwComponentType within a System as some sort of overall design model for software on both the *AUTOSAR classic platform* **and** the *AUTOSAR adaptive platform*.

This aspect, however, is not clarified so far nor is a restriction in place that prohibits AdaptiveApplicationSwComponentType to appear in the context of a System.

Later versions of this specification may fix the missing regulation.

Class	**AdaptiveApplicationSwComponentType**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This meta-class represents the ability to support the formal modeling of application software on the AUTOSAR adaptive platform. Consequently, it shall only be used on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=AdaptiveApplicationSwComponentTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
internalBehavior	AdaptiveSwcInternal Behavior	0..1	aggr	This aggregation represents the internal behavior of the AdaptiveApplicationSwComponentType for the AUTOSAR adaptive platform. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalBehavior.shortName, internal Behavior.variationPoint.shortLabel atp.Status=draft vh.latestBindingTime=preCompileTime

Table 3.1: AdaptiveApplicationSwComponentType

Class	**AdaptiveSwcInternalBehavior**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::AdaptiveInternalBehavior			
Note	This meta-class represents the ability to define an internal behavior of an AtomicSwComponentType used on the AUTOSAR adaptive platform. Please note that the model of internal behavior in this case, in stark contrast to the situation of the AUTOSAR classic platform, is very minimal. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
service Dependency	SwcService Dependency	*	aggr	This represents the collection of SwcService Dependencys owned by AdaptiveInternalBehavior. **Tags:**atp.Status=draft

Table 3.2: AdaptiveSwcInternalBehavior

3.3 Data Type

3.3.1 Overview

The specification of data types on the *AUTOSAR adaptive platform* follows the same pattern as the counterpart on the *AUTOSAR classic platform*: data types are defined on different levels of abstraction that complement each other.

In the context of this document, the focus is on the discussion of `ApplicationDataType`s and `CppImplementationDataType`s.

In general, most of the concepts regarding the definition of data types can be taken over from the existing specifications on the *AUTOSAR classic platform*.

However, some aspects are specific to the *AUTOSAR adaptive platform* and are consequently discussed in the scope of this document rather than the specification of the AUTOSAR Software Component Template [1].

One of the aspects that could be taken over from the *AUTOSAR classic platform* is the definition of initial values.

Although the utility of initial values is certainly limited on the *AUTOSAR adaptive platform*, there is an opportunity to utilize the definition of initial values in the context of the so-called `Field`s (see [TPS_MANI_01034]).

3.3.2 ApplicationDataType

The full range of the modeling of `ApplicationDataType`s that is supported on the *AUTOSAR classic platform* can directly be used on the *AUTOSAR adaptive platform* as well.

In addition to the `ApplicationDataType`s supported on the *AUTOSAR classic platform*, there are further `ApplicationDataType`s that – while in principle also available on the *AUTOSAR classic platform* – are primarily used on and designed for the *AUTOSAR adaptive platform*.

Class	***ApplicationDataType*** (abstract)
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes
Note	ApplicationDataType defines a data type from the application point of view. Especially it should be used whenever something "physical" is at stake. An ApplicationDataType represents a set of values as seen in the application model, such as measurement units. It does not consider implementation details such as bit-size, endianess, etc. It should be possible to model the application level aspects of a VFB system by using ApplicationData Types only.
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	**ApplicationDataType** (abstract)			
Subclasses	*ApplicationCompositeDataType*, ApplicationPrimitiveDataType			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.3: ApplicationDataType

3.3.2.1 String Data Type

While the handling of data types that represent textual strings is very similar with respect to the definition of `ApplicationDataType`s on the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform*, special regulations apply on the level of `CppImplementationDataType`s on the *AUTOSAR adaptive platform*.

For more information about the modeling of string data types on the level of `CppImplementationDataType` please refer to section 3.3.3.4.

For the sake of consistency, this chapter summarizes the modeling of `ApplicationDataType`s for the modeling of data types that represent textual strings as far as the *AUTOSAR adaptive platform* is concerned.

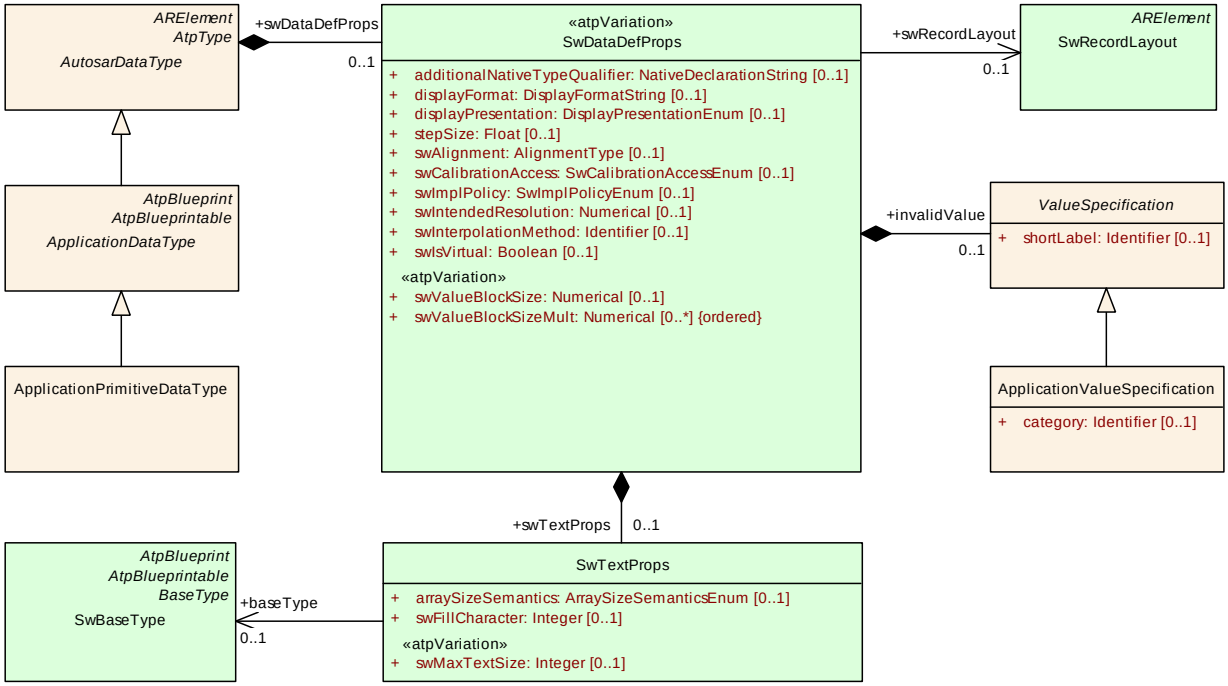


Figure 3.1: Specification of textual strings

The meta-classes used to define an `ApplicationPrimitiveDataType` of `category STRING` are summarized in Figure 3.1.

Please note that thanks to the usage of programming languages with richer data types than plain C, the implementation of an ApplicationPrimitiveDataType of category STRING on the *AUTOSAR adaptive platform* is predefined for a given *language binding*.

[TPS_MANI_01047]{DRAFT} **Existence of SwRecordLayout for an ApplicationPrimitiveDataType of category STRING** ⌈For the usage of an ApplicationPrimitiveDataType of category STRING on the *AUTOSAR adaptive platform*, the existence of ApplicationPrimitiveDataType.swDataDefProps.swRecordLayout shall be ignored.⌋*(RS_MANI_00016)*

Please note that [TPS_MANI_01047] intentionally does not forbid the existence of SwRecordLayout because the same ApplicationPrimitiveDataType of category STRING could rightfully be used **on both** the *AUTOSAR adaptive platform* and the *AUTOSAR classic platform*.

Class	**ApplicationPrimitiveDataType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	A primitive data type defines a set of allowed values. **Tags:**atp.recommendedPackage=ApplicationDataTypes			
Base	*ARElement*, *ARObject*, *ApplicationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.4: ApplicationPrimitiveDataType

Class	**SwTextProps**			
Package	M2::MSR::DataDictionary::DataDefProperties			
Note	This meta-class expresses particular properties applicable to strings in variables or calibration parameters.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls the semantics of the arraysize for the array representing the string in an Implementation DataType. It is there to support a safe conversion between ApplicationDatatype and ImplementationDatatype, even for variable length strings as required e.g. for Support of SAE J1939.
baseType	SwBaseType	0..1	ref	This is the base type of one character in the string. In particular this baseType denotes the intended encoding of the characters in the string on level of ApplicationData Type. **Tags:**xml.sequenceOffset=30

▽

△

Class	**SwTextProps**			
swFillCharacter	Integer	0..1	attr	Filler character for text parameter to pad up to the maximum length swMaxTextSize. The value will be interpreted according to the encoding specified in the associated base type of the data object, e.g. 0x30 (hex) represents the ASCII character zero as filler character and 0 (dec) represents an end of string as filler character. The usage of the fill character depends on the arraySize Semantics. **Tags:**xml.sequenceOffset=40
swMaxTextSize	Integer	0..1	attr	Specifies the maximum text size in characters. Note the size in bytes depends on the encoding in the corresponding baseType. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=20

Table 3.5: SwTextProps

3.3.2.2 Associative Map Data Type

[TPS_MANI_01027]{DRAFT} **Semantics of `ApplicationAssocMapDataType`** ⌈An `ApplicationAssocMapDataType` represents an associative data structure, i.e. a data structure where so-called *key*s (formalized as `ApplicationAssocMapDataType.key` that are in turn typed by an `ApplicationDataType`) are associated with *value*s (formalized as `ApplicationAssocMapDataType.value` that are also in turn typed by an `ApplicationDataType`).⌋*(RS_MANI_00016)*

[constr_3349]{DRAFT} **Usage of `ApplicationAssocMapDataType` is limited** ⌈The usage of an `ApplicationAssocMapDataType` is limited to the context of `AdaptiveApplicationSwComponentType`s and `CompositionSwComponentType`s defined in the context of an `Executable`, i.e. such a data type shall not be used on the *AUTOSAR classic platform*.⌋*()*

[constr_3349] is a formal approach to express that an `ApplicationAssocMapDataType` shall only be used on the *AUTOSAR adaptive platform*.

[TPS_MANI_01016]{DRAFT} **Category of `ApplicationAssocMapDataType`** ⌈The value `ApplicationAssocMapDataType.category` shall be set to `ASSOCIATIVE_MAP` for attribute.⌋*(RS_MANI_00016)*

Figure 3.2 depicts an example of the structure of an `ApplicationAssocMapDataType`.

As can be deduced from looking at Figure 3.2, the concept of an ApplicationDataType of category MAP shall not be confused with an ApplicationAssocMapDataType[1].

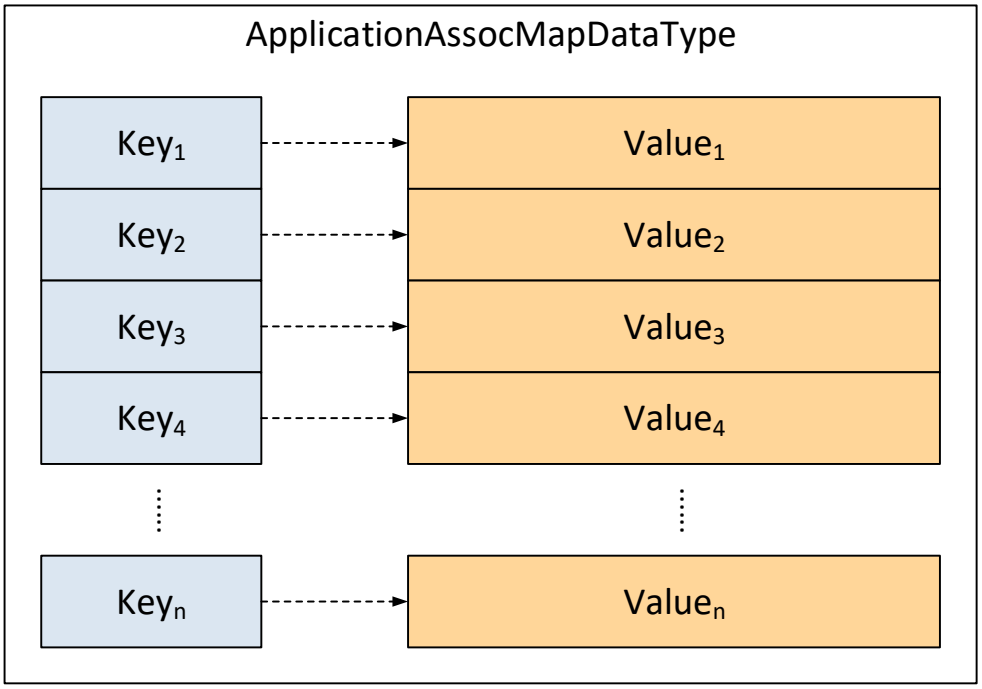


Figure 3.2: Example ApplicationAssocMapDataType on the *AUTOSAR adaptive platform*

There are a number of technical implications on the usage of an associative data structure at run-time, e.g. that the content of each *key* shall be unique within the context of the overall data structure.

On the other hand, it is totally no problem if content on the value-side contain duplicates, e.g. two unique keys are associated with values that have a completely identical content.

However, these aspects have no implication on the formal model of the ApplicationAssocMapDataType and are therefore not considered in this document.

The modeling of the ApplicationAssocMapDataType is somewhat minimalistic and motivated mainly be the fact that data types for both key and value need to be defined.

There is no assumption how the structure of an implementation of an associative map may look like. For example, in C++ (which is currently the only supported language

[1] On the other hand, both concepts of a “map” are justified in their respective “community” and choosing to name one of these very different in order so reduce overall potential confusion would probably not be applicable

binding on the *AUTOSAR adaptive platform*) the straightforward way to use an associative map is to utilize the container `ara::core::Map` (where the implementation is opaque to the client programmer).

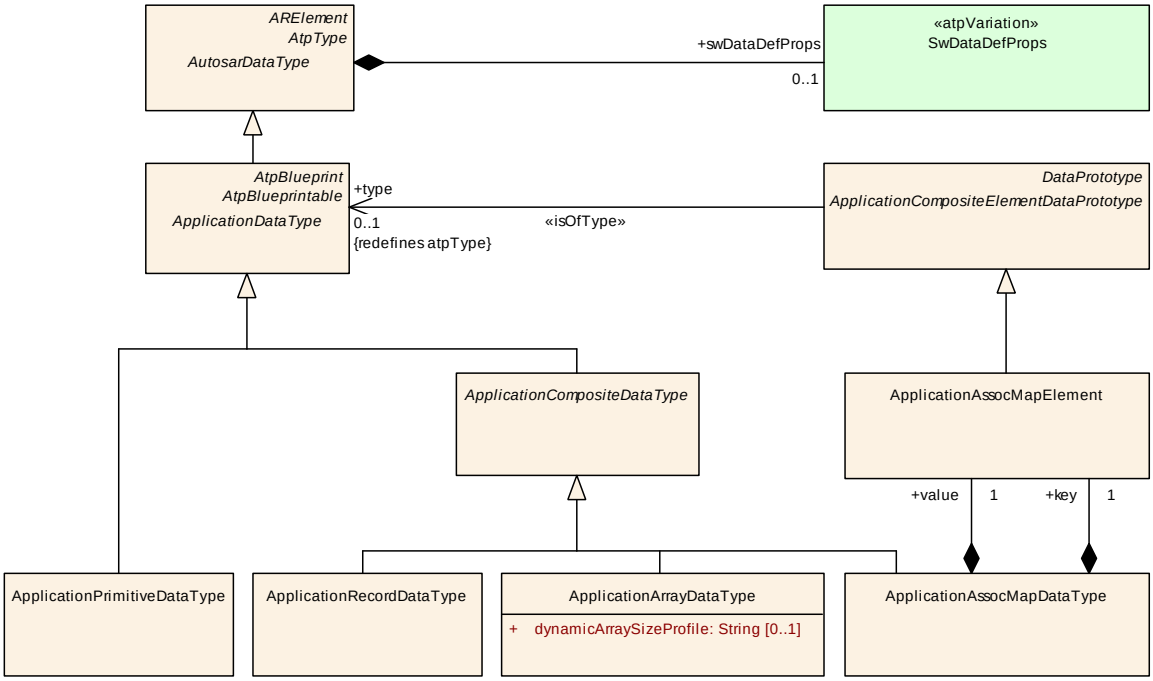


Figure 3.3: Formal model of `ApplicationAssocMapDataType`

Class	**ApplicationAssocMapDataType**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType			
Note	An application data type which is a map and consists of a key and a value **Tags:** atp.Status=draft atp.recommendedPackage=ApplicationDataTypes			
Base	*ARElement*, *ARObject*, *ApplicationCompositeDataType*, *ApplicationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
key	ApplicationAssocMapElement	1	aggr	Key element of the map that is used to uniquely identify the value of the map. **Tags:**atp.Status=draft
value	ApplicationAssocMapElement	1	aggr	Value element of the map that stores the content associated to a key. **Tags:**atp.Status=draft

Table 3.6: ApplicationAssocMapDataType

Class	**ApplicationAssocMapElement**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType			
Note	Describes the properties of the elements of an application map data type. **Tags:**atp.Status=draft			
Base	*ARObject*, *ApplicationCompositeElementDataPrototype*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.7: ApplicationAssocMapElement

Figure 3.4 contains a graphical representation of an example model for ApplicationAssocMapDataType.

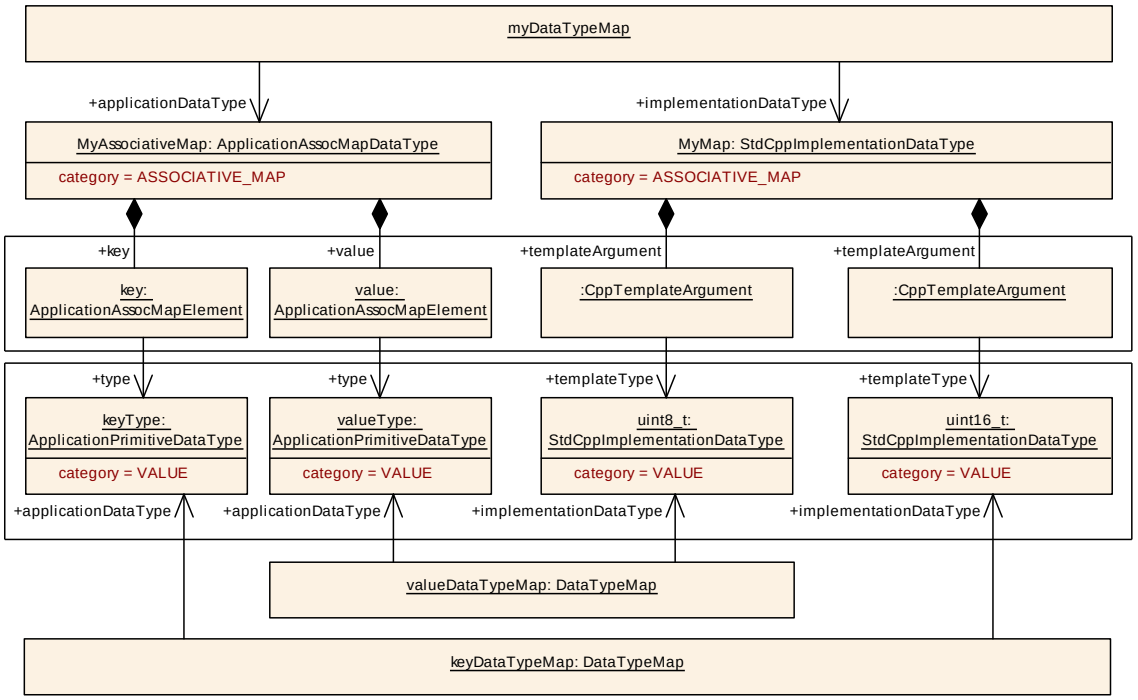


Figure 3.4: Example of the model of an associative map

Listing 3.1 provides the corresponding ARXML serialization of the example model of an ApplicationAssocMapDataType depicted in Figure 3.4.

Listing 3.1: Example for the definition of an ApplicationAssocMapDataType

```
<APPLICATION-ASSOC-MAP-DATA-TYPE>
  <SHORT-NAME>MyAssociativeMap</SHORT-NAME>
  <KEY>
    <SHORT-NAME>MyKey</SHORT-NAME>
    <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">keyType</TYPE-TREF>
  </KEY>
  <VALUE>
    <SHORT-NAME>MyValue</SHORT-NAME>
    <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">valueType</TYPE-TREF>
  </VALUE>
</APPLICATION-ASSOC-MAP-DATA-TYPE>
```

```
<APPLICATION-PRIMITIVE-DATA-TYPE>
  <SHORT-NAME>keyType</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
</APPLICATION-PRIMITIVE-DATA-TYPE>

<APPLICATION-PRIMITIVE-DATA-TYPE>
  <SHORT-NAME>valueType</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
</APPLICATION-PRIMITIVE-DATA-TYPE>
```

The initialization of an ApplicationAssocMapDataType, however, needs to be clarified because it would (using a combination of RecordValueSpecification and ArrayValueSpecification) in general be technically possible to define a number of differently structured ValueSpecifications that are semantically identical.

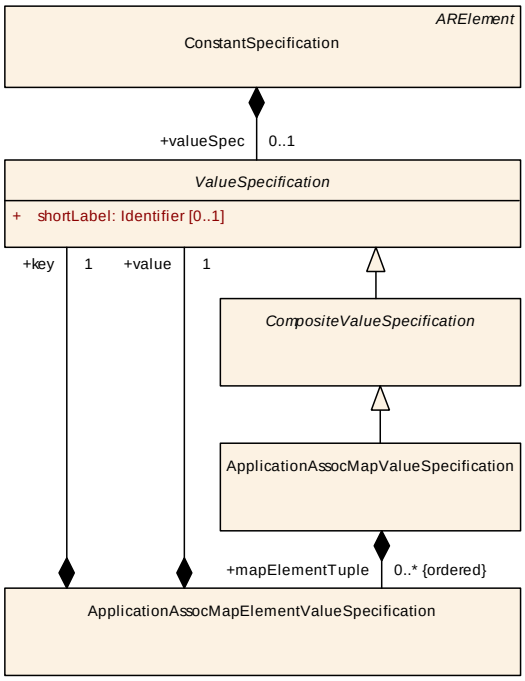


Figure 3.5: Formal model of the initialization of an ApplicationAssocMapDataType

In order to keep this element of uncertainty out of the AUTOSAR standard, the initialization of a DataPrototype typed by ApplicationAssocMapDataType is clarified by means of [constr_1488].

[constr_1488]{DRAFT} **Initialization of a DataPrototype typed by an ApplicationAssocMapDataType** ⌈A DataPrototype typed by an ApplicationAssocMapDataType shall only be initialized by an ApplicationAssocMapValueSpecification.⌋*()*

As already mentioned, there is a semantic requirement that the *key* elements of an *associative map* need to the unique in the context of one *associative map* container.

Obviously, the model has no influence on what happens at run-time. On the other hand, there is an implication onto the initialization of an ApplicationAssocMapDataType, see [constr_1489].

[constr_1489]{DRAFT} **Uniqueness of ApplicationAssocMapValueSpecification.mapElementTuple.key** ⌈The value of all mapElementTuple.key elements in the context of a given ApplicationAssocMapValueSpecification shall be unique.⌋*()*

Class	**ApplicationAssocMapValueSpecification**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType			
Note	This meta-class represents the ability to define the initialization of an ApplicationAssocMapDataType. **Tags:**atp.Status=draft			
Base	*ARObject*, *CompositeValueSpecification*, *ValueSpecification*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
mapElement Tuple (ordered)	ApplicationAssocMap ElementValue Specification	*	aggr	This aggregation represents the initial values for the elements of the ApplicationAssocMapValueSpecification. **Tags:**atp.Status=draft

Table 3.8: ApplicationAssocMapValueSpecification

Class	**ApplicationAssocMapElementValueSpecification**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType			
Note	This meta-class represents the ability to define the initialization of the elements of an ApplicationAssoc MapDataType. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
key	ValueSpecification	1	aggr	This aggregation represents the initialization of the key part of an AssociativeElementValueSpecification. **Tags:**atp.Status=draft
value	ValueSpecification	1	aggr	This aggregation represents the initialization of the value part of an AssociativeElementValueSpecification. **Tags:**atp.Status=draft

Table 3.9: ApplicationAssocMapElementValueSpecification

3.3.2.3 Attributes of SwDataDefProps

[constr_1478]{DRAFT} **SwDataDefProps applicable to ApplicationDataTypes exclusive to the *AUTOSAR adaptive platform*** ⌈A complete list of the SwDataDefProps and other attributes and their multiplicities which are allowed for a given category is shown in table 3.10.⌋*()*

A consequence of [constr_1478] is that the Table 3.10 shows only the values of category that are limited to the *AUTOSAR adaptive platform*. For all other values of category that are also supported on the *AUTOSAR classic platform* please refer to a similar table contained in the specification of the Software Component Template [1].

Attributes of SwDataDefProps	Root Elem.		Attribute Existence per Category
	ApplicationAssocMapDataType	ApplicationAssocMapElement	ASSOCIATIVE_MAP
additionalNativeTypeQualifier			
annotation	x	x	*
baseType			
compuMethod			
dataConstr			
displayFormat	x	x	0..1
implementationDataType			
invalidValue			
stepSize			
swAddrMethod			
swAlignment			
swBitRepresentation			
swCalibrationAccess			
swCalprmAxisSet			
swComparisonVariable			
swDataDependency			
swHostVariable			
swImplPolicy			
swIntendedResolution			
swInterpolationMethod			
swIsVirtual			
swPointerTargetProps			
swRecordLayout			
swRefreshTiming			
swTextProps			
swValueBlockSize			
unit			
valueAxisDataType			
Other Attributes below the Root Element			
key: ApplicationAssocMapElement	x		1
value: ApplicationAssocMapElement	x		1

Table 3.10: Allowed Attributes vs. category for ApplicationDataTypes

3.3.3 CppImplementationDataType

3.3.3.1 Overview

In the AUTOSAR standard, data types represent assets of paramount prominence for the entire development approach.

Therefore, AUTOSAR implements[2] a multi-level approach for the modeling of data types. One of the described levels, the so-called *Implementation Data Level* aims at a modeling on a level that could be described as "language binding" in the parlor of the *AUTOSAR adaptive platform*.

For the *AUTOSAR classic platform*, the *Implementation Data Level* has been addressed by the creation of the `ImplementationDataType` that specifically aims at covering the data type behavior of the C programming language.

In contrast to the *AUTOSAR classic platform*, the *AUTOSAR adaptive platform* currently does not foresee the usage of the C language and instead (at least for the foreseeable future) defines language binding to the C++ language.

It is therefore necessary to provide a modeling approach on the *Implementation Data Level* with a proper support for the capabilities of the C++ language.

While it would technically be feasible to extend the semantics of `ImplementationDataType` for a support of a C++ language binding this would significantly water down the clarity and expressiveness of `ImplementationDataType`[3].

It therefore seems reasonable to add a system of meta-classes that specifically supports the usage of data types with an intended binding to the C++ language.

[TPS_MANI_01166]{DRAFT} **Semantics of `CppImplementationDataType`** ⌈The abstract meta-class `CppImplementationDataType` supports the modeling of data types specifically tailored towards a support for a C++ language binding.⌋*(RS_MANI_00039)*

[TPS_MANI_03197]{DRAFT} **Semantics of `StdCppImplementationDataType`** ⌈Meta-class `StdCppImplementationDataType` supports the modeling of data types that will be mapped to C++ Standard Library features in the C++ language binding.⌋*(RS_MANI_00039)*

Please note that Structures (`category` = `STRUCTURE`) and type aliases (`category` = `TYPE_REFERENCE`) are also modeled as `StdCppImplementationDataType`s for simplification reasons.

[TPS_MANI_03198]{DRAFT} **Semantics of `CustomCppImplementationDataType`** ⌈Meta-class `CustomCppImplementationDataType` supports the

[2] As explained in [1]

[3] And even if it were possible to extend `ImplementationDataType` towards a more or less clean support for C++ it may happen that further language bindings are added to the *AUTOSAR adaptive platform* for which further and further extensions of `ImplementationDataType` would be required.

modeling of data types that will mapped to a custom implementation in the C++ language binding that is declared in the `headerFile`. ⌋*(RS_MANI_00039)*

Please note that the `category` values for a `CustomCppImplementationDataType` are restricted by [constr_1578].

This means that the modeling of primitive data types and strings is only possible with `StdCppImplementationDataType`s. The reason is that the serialization rules that are defined in AUTOSAR for SOME/IP and DDS are based on the defined types of the standard library.

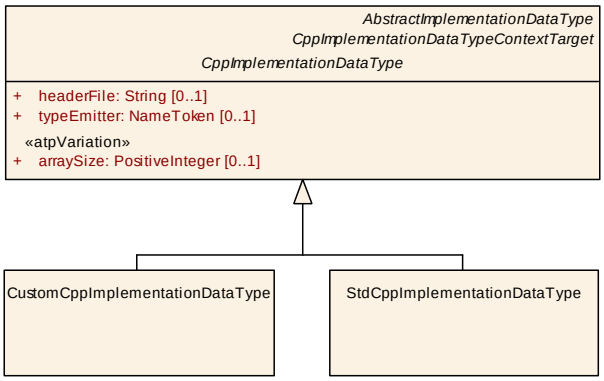


Figure 3.6: Specializations of `CppImplementationDataType`

Class	***CppImplementationDataType*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class represents the way to specify a reusable data type definition taken as a the basis for a C++ language binding **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	CustomCppImplementationDataType, StdCppImplementationDataType			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arraySize	PositiveInteger	0..1	attr	This attribute can be used to specify the array size if the enclosing CppImplementationDataType has array semantics. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
headerFile	String	0..1	attr	Configuration of the Header File with the custom class declaration.
namespace (ordered)	SymbolProps	*	aggr	This aggregation allows for the definition an own namespace for the enclosing CppImplementationData Type. **Tags:**atp.Status=draft
subElement (ordered)	CppImplementation DataTypeElement	*	aggr	This represents the collection of sub-elements of the enclosing CppImplementationDataType **Tags:**atp.Status=draft

△

Class	***CppImplementationDataType*** (abstract)			
template Argument (ordered)	CppTemplateArgument	*	aggr	This aggreation allows for the specification of properties of template arguments **Tags:**atp.Status=draft
typeEmitter	NameToken	0..1	attr	This attribute can be taken to control how the respective CppImplementationDataType is contributed to the language binding.
typeReference	CppImplementation DataType	0..1	ref	This reference shall be defined to define a type reference (a.k.a. typedef). **Tags:**atp.Status=draft

Table 3.11: CppImplementationDataType

Class	**StdCppImplementationDataType**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class represents the way to specify a data type definition that is taken as the basis for a C++ language binding to a C++ Standard Library feature. **Tags:** atp.Status=draft atp.recommendedPackage=CppImplementationDataTypes			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataType*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.12: StdCppImplementationDataType

Class	**CustomCppImplementationDataType**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class represents the way to specify a data type definition that is taken as the basis for a C++ language binding to a custom implementation that is declared in the configured header file. The Short Name of this CustomCppImplementationDataType defines the Class-Name of the custom implementation. **Tags:** atp.Status=draft atp.recommendedPackage=CppImplementationDataTypes			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataType*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.13: CustomCppImplementationDataType

[constr_1571]{DRAFT} **`CppImplementationDataType` is limited** ⌈The usage of a `CppImplementationDataType` is limited to the context of `AdaptiveApplicationSwComponentType`s and `CompositionSwComponentType`s defined in the context of an `Executable`.⌋*()*

[TPS_MANI_01167]{DRAFT} **`AbstractImplementationDataType`** ⌈Meta-class `CppImplementationDataType` inherits from abstract base class `AbstractImplementationDataType` in order to become a valid target for specific references from

other meta-classes that want to refer to "ImplementationDataType in general". *(RS_MANI_00039)*

Class	***AbstractImplementationDataType*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	This meta-class represents an abstract base class for different flavors of ImplementationDataType.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*CppImplementationDataType*, ImplementationDataType			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.14: AbstractImplementationDataType

A prominent example for the idea of referring to "ImplementationDataType in general" can be found in meta-class DataTypeMap. The intention behind the existence of DataTypeMap is to map an ApplicationDataType to either an ImplementationDataType or CppImplementationDataType.

By means of modeling the reference DataTypeMap.implementationDataType as a reference to AbstractImplementationDataType both options are possible in a single role.

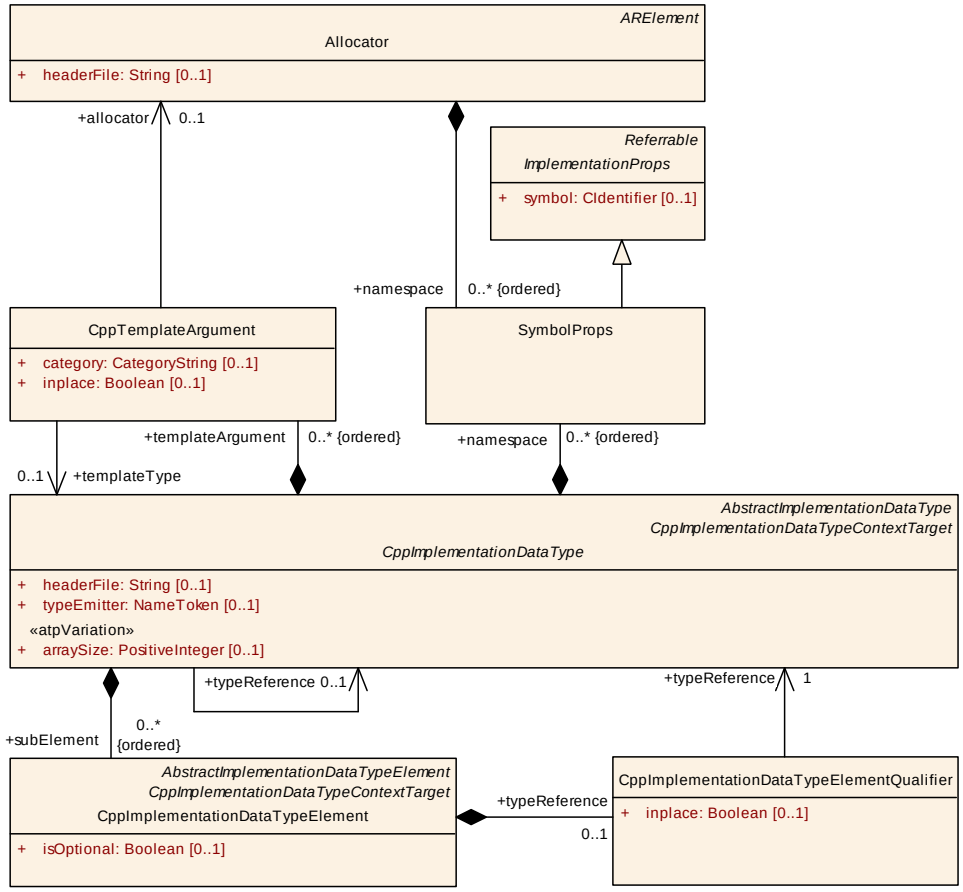


Figure 3.7: CppImplementationDataType overview

In contrast to the C language, C++ supports the definition of namespaces in programs. This feature is also cleared for development on the *AUTOSAR adaptive platform* and therefore needs to be represented in the modeling approach.

[TPS_MANI_01168]{DRAFT} **Specification of a namespace for a CppImplementationDataType** ⌈The ability to define a namespace for a CppImplementationDataType is expressed by means of the aggregation of SymbolProps at CppImplementationDataType in the role namespace.⌋*(RS_MANI_00039)*

[constr_3443]{DRAFT} **Specification of a namespace for a StdCppImplementationDataType** ⌈The definition of a namespace for a StdCppImplementationDataType of category VALUE is not allowed. For this value of category the std namespace is already assumed by the usage of the StdCppImplementationDataType.⌋*()*

[TPS_MANI_01309]{DRAFT} **Semantics of attribute CppImplementationDataType.headerFile** ⌈The attribute CppImplementationDataType.headerFile shall be used to specify the name of the corresponding header file in two cases:

- A CustomCppImplementationDataType shall set the value of the attribute to the name of the header file that defines the C++ code for the CustomCppImplementationDataType.
- A platform data type (modeled as a StdCppImplementationDataType) shall set the attribute to the name of the applicable header file (e.g. "cstdint") from the C++ standard library.

⌋*()*

[constr_1743]{DRAFT} **CppImplementationDataType.headerFile vs. CppImplementationDataType.typeEmitter** ⌈The two attributes CppImplementationDataType.headerFile and CppImplementationDataType.typeEmitter shall always be used mutually exclusive.

In other words, a subclass of CppImplementationDataType shall either use headerFile or typeEmitter. The simultaneous usage of both attributes is not supported.⌋*()*

[TPS_MANI_01176]{DRAFT} **Standardized value for attribute CppImplementationDataType.typeEmitter** ⌈The AUTOSAR Standard reserves the following value for attribute CppImplementationDataType.typeEmitter:

- TYPE_EMITTER_ARA

⌋*(RS_MANI_00039)*

[TPS_MANI_01177]{DRAFT} **Semantics of attribute CppImplementationDataType.typeEmitter** ⌈The following set of rules applies for the usage of the attribute CppImplementationDataType.typeEmitter:

- If the attribute typeEmitter is set to the value TYPE_EMITTER_ARA, the ARA generator shall generate the corresponding data type definition.

- If the attribute typeEmitter is set to any value other than TYPE_EMITTER_ARA, the ARA generator shall silently **not** generate the corresponding data type definition.

⌋(*RS_MANI_00039*)

In the context of [TPS_MANI_01177], [TPS_MANI_01309] and [constr_1743] apply.

[TPS_MANI_01212]{DRAFT} **Usage of attribute typeEmitter in the context of a CustomCppImplementationDataType** ⌈Attribute typeEmitter does not have to be used in the context of a CustomCppImplementationDataType. If the typeEmitter is used regardless then the value of the attribute shall be set to the name of the header file that contains the language binding of the respective CustomCppImplementationDataType.⌋(*RS_MANI_00039*)

[TPS_MANI_01169]{DRAFT} **Support for template data types** ⌈Meta-class CppImplementationDataType supports the usage of templates for the definition of data types in C++ programs by means of the reference CppImplementationDataType.templateArgument.

The order of arguments in templates is significant, therefore templateArgument is modeled as an **ordered** collection.⌋(*RS_MANI_00039*)

[TPS_MANI_01174]{DRAFT} **Semantics of reference in the role CppTemplateArgument.templateType** ⌈Attribute CppTemplateArgument.templateType specifies the data type to be filled in the respective position of the template in the language binding.⌋(*RS_MANI_00039*)

[TPS_MANI_01175]{DRAFT} **Semantics of reference in the role CppTemplateArgument.allocator** ⌈Attribute CppTemplateArgument.allocator specifies the behavior of an allocator class to be filled in the respective position of the template in the language binding.⌋(*RS_MANI_00039*)

[constr_1576]{DRAFT} **Existence of CppTemplateArgument.templateType vs. CppTemplateArgument.allocator** ⌈For any given CppTemplateArgument, **at most one of** the references

- CppTemplateArgument.templateType or
- CppTemplateArgument.allocator

may exist.⌋()

[TPS_MANI_01201]{DRAFT} **Standardized values for attribute CppTemplateArgument.category** ⌈AUTOSAR reserves the following values for attribute CppTemplateArgument.category:

ASSOC_MAP_KEY : the specific CppTemplateArgument represents the *key* data-type of an associative map.

ASSOC_MAP_VALUE : the specific CppTemplateArgument represents the *value* data-type of an associative map.

⌋(*RS_MANI_00039*)

Class	**CppTemplateArgument**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class has the ability to define properties for template arguments. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allocator	Allocator	0..1	ref	This reference identifies the applicable allocator. **Tags:**atp.Status=draft
category	CategoryString	0..1	attr	This attribute shall be used to contribute further clarification regarding the semantics of the enclosing Cpp TemplateArgument.
inplace	Boolean	0..1	attr	This attribute specifies whether the shortName of the referenced templateType is used in the code generation and the type declaration is defined outside of the enclosing CppImplementationDataType (true) or whether the type definition is embedded inside of the enclosing CppImplementationDataType and the shortName is ignored (false).
templateType	CppImplementation DataType	0..1	ref	This reference identifies the data type of the specific template argument required for the language binding. **Tags:**atp.Status=draft

Table 3.15: CppTemplateArgument

[TPS_MANI_01171]{DRAFT} **Modeling of structured data types** ⌈Meta-class `CppImplementationDataType` supports the creation of nested data types by means of the aggregation of `CppImplementationDataTypeElement` in the role `subElement`.

Because the order of sub-elements in a structured data type is significant the aggregation `subElement` is modeled as an **ordered** collection.⌋(*RS_MANI_00039*)

Please note that although the modeling of structures is formally done by way of using `CppImplementationDataType` it is actually only possible to use `StdCppImplementationDataType` for this purpose (see [constr_1578]).

Class	**CppImplementationDataTypeElement**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	Declares a data object which is locally aggregated. Such an element can only be used within the scope where it is aggregated. A CppImplementationDataTypeElement is used to represent an element of a structure, defining its type. **Tags:**atp.Status=draft			
Base	*ARObject*, *AbstractImplementationDataTypeElement*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	CppImplementationDataTypeElement			
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing CppImplementationDataTypeElement as optional. This means the that, at runtime, the Cpp ImplementationDataTypeElement may or may not have a valid value and shall therefore be ignored. The underlying runtime software provides means to set the CppImplementationDataTypeElement as not valid at the sending end of a communication and determine its validity at the receiving end.
typeReference	CppImplementation DataTypeElement Qualifier	0..1	aggr	This aggregation defines the type of the Cpp ImplementationDataTypeElement and determines whether in C++ the CppImplementationDataTypeElement is defined inside or outside of the enclosing Cpp ImplementationDataType. **Tags:**atp.Status=draft

Table 3.16: CppImplementationDataTypeElement

Please note that there is no intention to support a “mixed” modeling of structured data types such that the resulting data type on C++ level would be composed of data types that are native to C++ and data types from the C subsystem.

While this would technically be possible on code level it would impose a huge effort on modeling level and the consensus is that there is no real use case for such a "mixed" data type.

The C++ data type system can, as far as the implementation of the *AUTOSAR adaptive platform* is concerned, fully replace the “legacy” C data types in C++.

[constr_1572]{DRAFT} **Usage of `SwDataDefProps.implementationDataType` within a `CppImplementationDataType`** ⌈Within the scope of a `CppImplementationDataType` the reference `CppImplementationDataType.swDataDefProps.implementationDataType` **shall not exist**.⌋*()*

This aspect is also expressed in a more general form by [constr_1579].

As a consequence of [constr_1572], type-references have to be done differently on the *AUTOSAR adaptive platform*. For this purpose dedicated references are available.

[TPS_MANI_01172]{DRAFT} **Description of type references in the scope of `CppImplementationDataType`** ⌈The reference `CppImplementationDataType.typeReference` can be used to create a type reference from the enclosing `CppImplementationDataType` to another `CppImplementationDataType`.⌋*(RS_MANI_00039)*

[TPS_MANI_01173]{DRAFT} **Description of type references in the scope of `CppImplementationDataTypeElement`** ⌈`CppImplementationDataTypeElement.typeReference` can be used to create a reference to the `CppImplementationDataType` that shall apply for the enclosing `CppImplementationDataTypeElement`.⌋*(RS_MANI_00039)*

Please note that the CppImplementationDataTypeElement.typeReference is realized as an Association Class that allows to add the inplace attribute to the typeReference.

Class	**CppImplementationDataTypeElementQualifier**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This element qualifies the typeReference of the CppImplementationDataTypeElement to the Cpp ImplementationDataType. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
inplace	Boolean	0..1	attr	This attribute defines whether the member type of the CppImplementationDataTypeElement in C++ is an embedded type element inside of the enclosing struct (true) or whether the type declaration is defined outside of the struct.
typeReference	CppImplementation DataType	1	ref	This reference defines a type reference. **Tags:**atp.Status=draft

Table 3.17: CppImplementationDataTypeElementQualifier

[TPS_MANI_03196]{DRAFT} **Semantics of CppImplementationDataTypeElementQualifier.inplace attribute** ⌈The CppImplementationDataTypeElementQualifier.inplace attribute defines whether the data type of the CppImplementationDataTypeElement in the C++ language binding is derived from the name or the properties of the referenced CppImplementationDataType.

Specifically, the following rules shall apply:

- if CppImplementationDataTypeElement.typeReference.inplace is set to False then the **shortName** of the CppImplementationDataType referenced in the role CppImplementationDataTypeElement.typeReference. typeReference shall be used in the C++ language binding.
- if CppImplementationDataTypeElement.typeReference.inplace is set to True then only the **properties** of the CppImplementationDataType referenced in the role CppImplementationDataTypeElement.typeReference. typeReference shall be used in the C++ language binding and the shortName is ignored.

⌋*(RS_MANI_00039)*

Please note that Figure 3.13 shows an example of a Structure where the typeReference of one subElement is classified as inplace.

[constr_1659]{DRAFT} **Restriction for the usage of CppImplementationDataTypeElementQualifier.inplace** ⌈The attribute CppImplementationDataTypeElementQualifier.inplace shall only exist if the target referenced in the role CppImplementationDataTypeElementQualifier.typeReference is an StdCppImplementationDataType that has the attribute category set to either of the values

- ARRAY
- VECTOR
- ASSOCIATIVE_MAP
- VARIANT
- STRUCTURE
- STRING
- TYPE_REFERENCE, if the CppImplementationDataType refers to a CompuMethod of category TEXTTABLE

⌋()

Rationale for the existence of [constr_1659]: by application of the exclusion principle, there are three cases where attribute CppImplementationDataTypeElementQualifier.inplace shall not exist:

- StdCppImplementationDataType of category VALUE
- CustomCppImplementationDataType
- CppImplementationDataType of category TYPE_REFERENCE, unless the CppImplementationDataType refers to a CompuMethod of category TEXTTABLE

Neither of them can be used as a target of CppImplementationDataTypeElementQualifier.typeReference where CppImplementationDataTypeElementQualifier.inplace is set to True because in these cases there is already a valid name that is directly usable for the language binding and a possible indirection via a using clause would obviously require an additional name that is not available from the model.

After all, the motivation for the definition of a TYPE_REFERENCE is the direct opposite of the motivation behind using the attribute CppImplementationDataTypeElementQualifier.inplace to control the language binding. Therefore, this case is also excluded.

[TPS_MANI_03201]{DRAFT} **Semantics of CppTemplateArgument.inplace attribute** ⌈The CppTemplateArgument.inplace attribute defines whether the data type that is referenced by the templateType in the C++ language binding is derived from the name or the properties of the referenced CppImplementationDataType.

Specifically, the following rules shall apply:

- if CppTemplateArgument.inplace is set to False then the **shortName** of the CppImplementationDataType referenced in the role CppTemplateArgument.templateType shall be used in the C++ language binding.

- if CppTemplateArgument.inplace is set to True then only the **properties** of the CppImplementationDataType referenced in the role CppTemplateArgument.templateType shall be used in the C++ language binding and the shortName is ignored.

⌋(*RS_MANI_00039*)

[constr_1660]{DRAFT} **Restriction for the usage of CppTemplateArgument.inplace** ⌈The attribute CppTemplateArgument.inplace shall only exist if the target referenced in the role CppTemplateArgument.templateType is an StdCppImplementationDataType that has the attribute category set to either of the values

- ARRAY
- VECTOR
- ASSOCIATIVE_MAP
- VARIANT
- STRUCTURE
- STRING

⌋()

Rationale for the existence of [constr_1660]: by application of the exclusion principle, there are three cases where attribute CppTemplateArgument.inplace shall not exist:

- StdCppImplementationDataType of category VALUE
- CustomCppImplementationDataType
- CppImplementationDataType of category TYPE_REFERENCE

Neither of them can be used as a target of CppTemplateArgument.templateType where CppTemplateArgument.inplace is set to True because in these cases there is already a valid name that is directly usable for the language binding and a possible indirection via a using clause would obviously require an additional name that is not available from the model.

After all, the motivation for the definition of a TYPE_REFERENCE is the direct opposite of the motivation behind using the attribute CppTemplateArgument.inplace to control the language binding. Therefore, this case is also excluded.

Please note that the question of the value of attribute CppTemplateArgument.inplace for the case of CppTemplateArgument.templateType referring to StdCppImplementationDataType of category STRUCTURE is regulated by [constr_3462].

[constr_1708]{DRAFT} **Combination of `CppImplementationDataTypeElement.isOptional` and `CppImplementationDataTypeElementQualifier.inplace`** ⌈If a `CppImplementationDataTypeElement` is typed by a `CppImplementationDataType` of `category STRUCTURE` then the combination of attribute `CppImplementationDataTypeElement.isOptional` set to True and `CppImplementationDataTypeElement.typeReference.inplace` set to `True` is not allowed.⌋*()*

Rationale for the existence of [constr_1708]: the "optional" semantics is implemented via a template and it is not possible to pass an "inplace" structure as a template argument.

[constr_3462]{DRAFT} **`CppTemplateArgument.templateType` reference to `StdCppImplementationDataType` of `category STRUCTURE` and the `inplace` flag** ⌈`CppTemplateArgument.templateType` that points to a `StdCppImplementationDataType` of `category STRUCTURE` shall have the `inplace` attribute set to **false**.⌋*()*

The reason for [constr_3462] is that the usage of an unnamed struct as template argument is not permitted by ISO C++11/14/17.

[constr_3446]{DRAFT} **`CppTemplateArgument` with `allocator` reference and the `inplace` flag** ⌈A `CppTemplateArgument` that points with an `allocator` reference to an `Allocator` shall not have the `inplace` flag set to a value.⌋*()*

Class	**Allocator**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class represents the ability to take influence on the way objects are allocated in memory, for example it can be controlled whether an objects is allocated on the heap or on the stack. **Tags:** atp.Status=draft atp.recommendedPackage=Allocators			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
headerFile	String	0..1	attr	Configuration of the Header File with the custom class declaration
namespace (ordered)	SymbolProps	*	aggr	This aggregation allows for the definition of a namespace of an Allocator. **Tags:**atp.Status=draft

Table 3.18: Allocator

[TPS_MANI_01100]{DRAFT} **Semantics of `Allocator`** ⌈Meta-class `Allocator` carries the ability to define the properties of an allocation of memory. The general approach for memory allocation is expressed by means of the attribute `category`.

The following values of `Allocator.category` are standardized by AUTOSAR:

- `MAX_SIZE_HEAP`: when using this allocator there is the intention to allocate a fixed-size chunk on the heap. This allocator adds the ability to define a maximum

number of elements to the semantics of the default allocator of `ara::core::Vector`.

- `MAX_SIZE_STACK`: when using this allocator there is the intention to allocate a fixed-size chunk on the stack. Memory on the stack always needs to be constrained in terms of the maximum size. In other words, there is hardly any case where an unbounded amount of memory should be allocated on the stack.
- `MAX_SIZE_DATASEGMENT`: when using this allocator there is the intention to allocate a fixed-size chunk in the data segment.

⌋(*RS_MANI_00016*)

[constr_1578]{DRAFT} **applicable data categories** ⌈Table 3.19 defines the applicable `category`s vs. meta-class.⌋()

Category	Applicable to ...								Description
	ApplicationArrayDataType	ApplicationRecordDataType	ApplicationPrimitiveDataType	ApplicationRecordElement	ApplicationArrayElement	ApplicationValueSpecification	StdCppImplementationDataType	CustomCppImplementationDataType	
VALUE			x	x	x	x	x		Contains a single value. See also [TPS_MANI_03192].
TYPE_REFERENCE							x		The element is defined via reference to another data type (via CppImplementationDataType.typeReference.
STRUCTURE		x		x	x		x		Holds one or several further elements which can have different AutosarDataTypes. See also [TPS_MANI_03180].
VARIANT							x	x	Can hold values of different data types. It is similar to STRUCTURE except that all of its members start at the same location in memory. A VARIANT data prototype can contain only one of its elements at a time and represents a type-safe union. The size of the VARIANT is at least the size of the largest member. See also [TPS_MANI_03189].
ARRAY	x			x	x		x	x	A fixed-sized array of sub-elements of the same data type. See also [TPS_MANI_03169].
VECTOR							x	x	An array of elements of the same data type that is able to grow at run-time. See also [TPS_MANI_03174].
ASSOCIATIVE_MAP							x	x	An associative array of key-value pairs. See also [TPS_MANI_03183].
STRING			x	x	x	x	x		Contains a text string. See also [TPS_MANI_03178].
BOOLEAN			x	x	x	x			Contains one boolean state. Depending on the CPU direct addressing of single bits may not be available. So a byte or a word can be used to store only one logical state.

Table 3.19: Usage of `category` for Data Types

3.3.3.2 Attributes of SwDataDefProps

[constr_1579]{DRAFT} **SwDataDefProps applicable to CppImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*** ⌈A complete list of the SwDataDefProps and other attributes and their multiplicities which are allowed for a given category is shown in table 3.20.⌋*()*

A consequence of [constr_1578] is that the Table 3.20 shows only the values of category that are limited to the *AUTOSAR adaptive platform*. For all other values of category that are also supported on the *AUTOSAR classic platform* please refer to a similar table contained in the specification of the Software Component Template [1].

Attributes of SwDataDefProps	Root Element	Attribute Existence per Category							
	CppImplementationDataType	VALUE	TYPE_REFERENCE	STRUCTURE	VARIANT	ARRAY	VECTOR	ASSOCIATIVE_MAP	STRING
additionalNativeTypeQualifier									
annotation	x	*	*	*	*	*	*	*	*
baseType									
compuMethod	x		0..1						
dataConstr.dataConstrRule.physConstrs	x		d/c			d/c	d/c		
dataConstr.dataConstrRule.internalConstrs	x		0..1			0..1	0..1		
displayFormat	x	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1
implementationDataType									
invalidValue	x		0..1						0..1
stepSize									
swAddrMethod									
swAlignment									
swBitRepresentation									
swCalibrationAccess									
swCalprmAxisSet									
swComparisonVariable									
swDataDependency									
swHostVariable									
swImplPolicy									
swIntendedResolution									
swInterpolationMethod									
swIsVirtual									
swPointerTargetProps									

▽

△

Attributes of SwDataDefProps	Root Element	Attribute Existence per Category							
	CppImplementationDataType	VALUE	TYPE_REFERENCE	STRUCTURE	VARIANT	ARRAY	VECTOR	ASSOCIATIVE_MAP	STRING
swPointerTargetProps.swDataDefProps									
swRecordLayout									
swRefreshTiming	x	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1
swTextProps									
swValueBlockSize									
unit									
valueAxisDataType									
Other Attributes									
subElement: CppImplementationDataTypeElement	x			1..*					
templateArgument	x				1..*	1	1..*	2..*	0..1
typeReference	x		1						

Table 3.20: Allowed Attributes vs. category for CppImplementationDataType

The invalidValue is applicable to Primitive Data Types and defines one specific value (in the range of that Primitive Data Type) which indicates that the respective value is not valid.

A typical use case is a composite data type that contains the values of all 4 wheel speeds. If one of the wheel speed sensors fails, and is no longer able to provide useful data, it does still make sense to provide the other 3 wheel speed values.

In such a scenario the one wheel speed value would then be set to the invalidValue. The receivers are able to check for each individual element of the data composition whether the value corresponds to the invalidValue and take corresponding actions.

[constr_3569]{DRAFT} **Applicability of attribute invalidValue on CppImplementationDataType of category TYPE_REFERENCE** ⌈If a CppImplementationDataType of category TYPE_REFERENCE has an invalidValue defined, then the referenced CppImplementationDataType (via typeReference) shall eventually be of category VALUE.⌋*()*

3.3.3.3 Primitive Data Types

[TPS_MANI_03192]{DRAFT} **CppImplementationDataType of category VALUE** ⌈The primitive data types like Boolean, fixed-width integer data types and

floating-point data types are described as CppImplementationDataTypes of category VALUE.⌋(*RS_MANI_00039*)

[TPS_MANI_03193]{DRAFT} **CppImplementationDataType of category TYPE_REFERENCE** ⌈The definition of a CppImplementationDataType of category TYPE_REFERENCE creates an alias for another CppImplementationDataType that is referenced by the typeReference.⌋(*RS_MANI_00039*)

3.3.3.4 String Data Type

[TPS_MANI_03178]{DRAFT} **StdCppImplementationDataType of category STRING** ⌈A StdCppImplementationDataType of category STRING represents a container data type for a sequence of characters.

AUTOSAR demands that the C++ binding of a StdCppImplementationDataType of category STRING is implemented by a ara::core::String.⌋(*RS_MANI_00039*)

[constr_1674]{DRAFT} **Supported encoding of StdCppImplementationDataType of category STRING** ⌈On the level of the meta-model (and, by extension, the language binding), the only supported encoding of StdCppImplementationDataType of category STRING is UTF-8.⌋()

Please note that it is nonetheless possible to use a different encoding, e.g. UTF-16 on the level of a SOME/IP message. This behavior can be configured by means of ApSomeipTransformationProps. As a consequence, a transcoding may have to be applied between the representation of a string on the wire and in the software.

[TPS_MANI_03179]{DRAFT} **C++ language binding of StdCppImplementationDataTypes of category STRING** ⌈A CppImplementationDataType of category STRING shall be implemented as ara::core::String.⌋(*RS_MANI_00039*)

The formulation of [TPS_MANI_03179] leaves room for potential later extensions towards the support for other storage formats.

The example depicted in Figure 3.8 contains the definition of both an ApplicationDataType as well as the definition of the corresponding CppImplementationDataType.

The latter obviously becomes significantly lighter to model thanks to the restriction that, as far as the C++ language binding is concerned, a CppImplementationDataType of category STRING shall only be implemented on the basis of an ara::core::String.

Another aspect of the example in Figure 3.8 is that it defines the intended encoding of the modeled data type in the scope of the ApplicationPrimitiveDataType.

[TPS_MANI_03188]{DRAFT} **Usage of an Allocator for a StdCppImplementationDataType of category STRING** ⌈A StdCppImplementationDataType of

category STRING is allowed to aggregate a CppTemplateArgument that refers to an Allocator with the allocator reference.⌋(RS_MANI_00039)

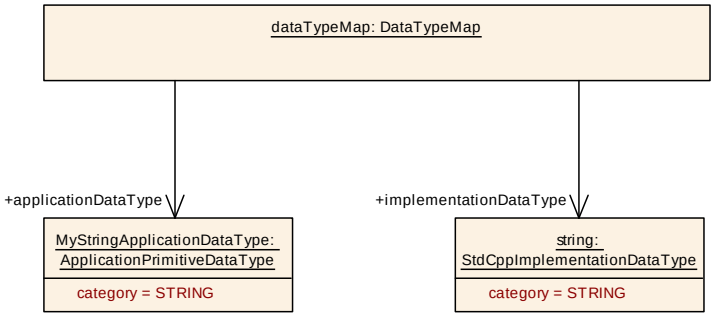


Figure 3.8: Example of the model of a string with UTF-8 encoding

3.3.3.5 Array Data Type

[TPS_MANI_03169]{DRAFT} **CppImplementationDataType with fixed size array semantics** ⌈A CppImplementationDataType of category ARRAY represents a container data type that encapsulates fixed size arrays.⌋(RS_MANI_00039)

[TPS_MANI_03170]{DRAFT} **CppImplementationDataType of category ARRAY** ⌈For a C++ binding, a CppImplementationDataType of category ARRAY can be implemented as

- an ara::core::Array if StdCppImplementationDataType subclass is used for modeling or as
- an array type in a custom namespace (e.g. my::array) if CustomCppImplementationDataType subclass is used (provided that the type in the custom namespace can be configured with the available modeling capabilities).

⌋(RS_MANI_00039)

[TPS_MANI_03171]{DRAFT} **Value type of a CppImplementationDataType of category ARRAY** ⌈The type of elements contained in a CppImplementationDataType of category ARRAY is defined by the aggregated templateArgument and the corresponding templateType that defines the data type of the CppTemplateArgument.⌋(RS_MANI_00039)

[constr_3433]{DRAFT} **Aggregation of templateArguments for an ARRAY** ⌈CppImplementationDataType of category ARRAY that boils down to ara::core::Array shall aggregate exactly one templateArgument that defines the type of elements contained in the CppImplementationDataType of category ARRAY.⌋()

[TPS_MANI_03172]{DRAFT} **Size of a `CppImplementationDataType` of category `ARRAY`** ⌈The primitive attribute `arraySize` of a `CppImplementationDataType` of `category ARRAY` shall be used to define the size of the array.⌋*(RS_MANI_00039)*

Figure 3.9 shows an example of an one-dimensional array of uint16 elements with `arraySize` = 5.

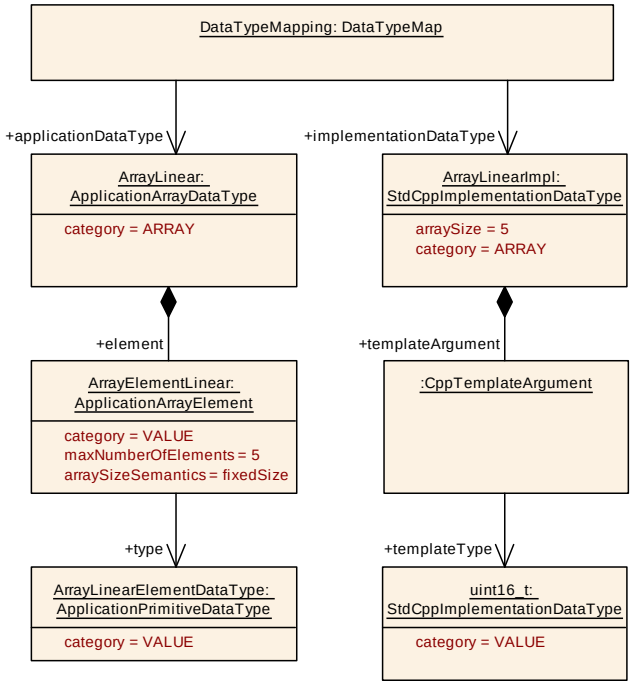


Figure 3.9: Example of the model of a one-dimensional array

[TPS_MANI_03173]{DRAFT} **Definition of a multidimensional Array** ⌈A multidimensional `CppImplementationDataType` of `category ARRAY` contains nested `CppImplementationDataType`s of `category ARRAY`.

The `CppImplementationDataType` of `category ARRAY` that represents the outer array will refer to a `CppImplementationDataType` of `category ARRAY` that represents the inner array via the aggregated `templateArgument`. Such a definition describes a two-dimensional Array; consequently a type with more dimensions is described by just nesting more `CppImplementationDataType`s of `category ARRAY`.

The array element itself is specified by the innermost `CppImplementationDataType` with `category` different from `ARRAY`.⌋*(RS_MANI_00039)*

Figure 3.10 shows an example of a multidimensional array where a `CppImplementationDataType` of `category ARRAY` with `arraySize` = 5 has a `templateArgument` that points to the inner `CppImplementationDataType` of `category ARRAY` in the role `templateType`.

The inner `CppImplementationDataType` has a `templateArgument` that finally points with the `templateType` reference to a primitive type.

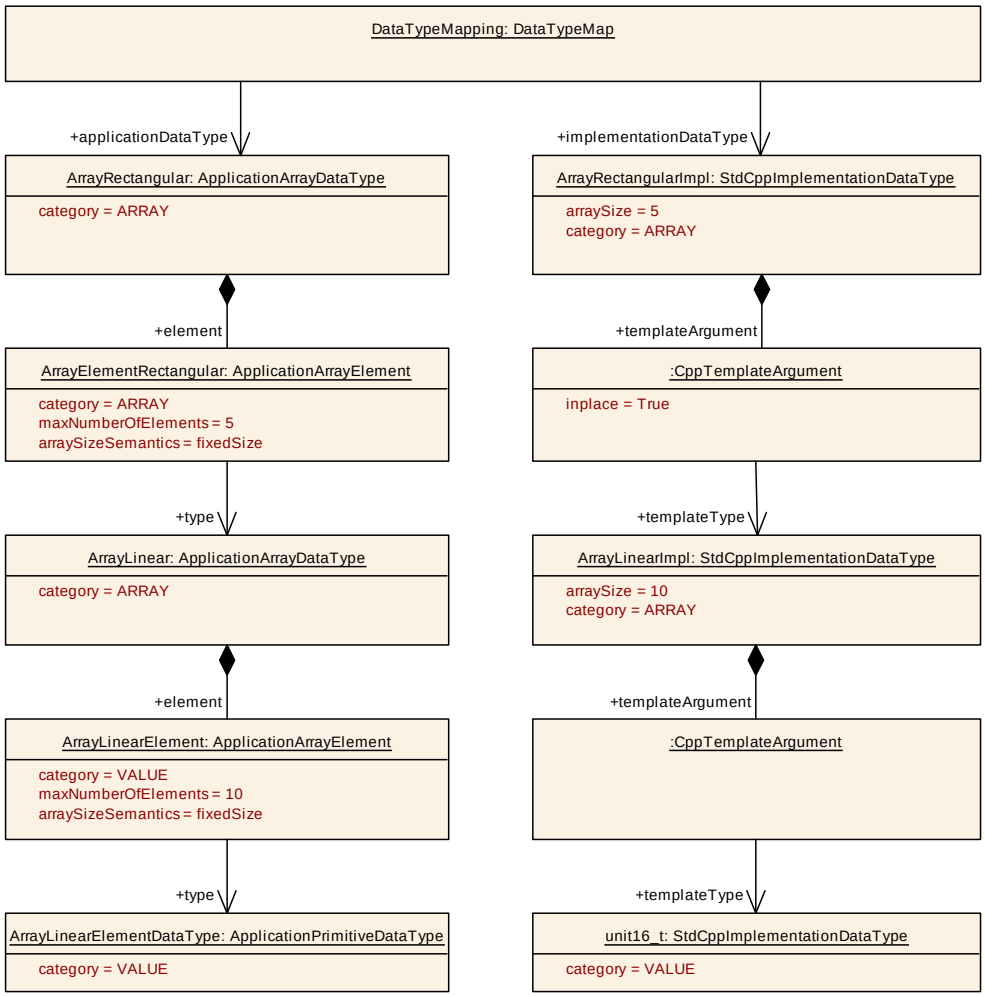


Figure 3.10: Example of the model of a multidimensional array

3.3.3.6 Vector Data Type

[TPS_MANI_03174]{DRAFT} **CppImplementationDataType with variable size array semantics** ⌈A `CppImplementationDataType` of `category VECTOR` represents a container data type that encapsulates variable size arrays.⌋*(RS_MANI_00039)*

[TPS_MANI_03175]{DRAFT} **CppImplementationDataType of category VECTOR** ⌈For a C++ binding, a `CppImplementationDataType` of `category VECTOR` can be implemented as

- an `ara::core::Vector` if `StdCppImplementationDataType` subclass is used or as
- a vector type in a custom namespace (e.g. `my::vector`) if `CustomCppImplementationDataType` subclass is used (provided that the type in the custom namespace can be configured with the available modeling capabilities).

⌋*(RS_MANI_00039)*

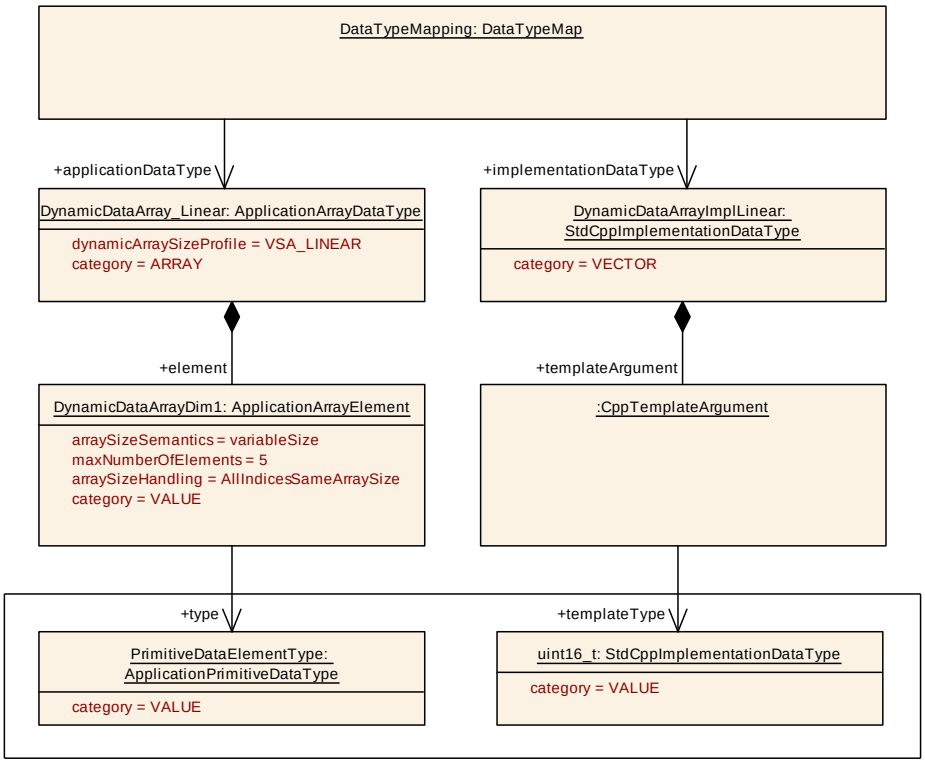


Figure 3.11: Example of the model of a one-dimensional vector

[TPS_MANI_03176]{DRAFT} **Value type of a `CppImplementationDataType` of category `VECTOR`** ⌈The type of elements contained in a `CppImplementationDataType` of `category VECTOR` is defined by the aggregated `templateArgument` and the corresponding `templateType` that defines the data type of the `CppTemplateArgument`.⌋*(RS_MANI_00039)*C

[constr_3434]{DRAFT} **Aggregation of `templateArgument`s for a `VECTOR`** ⌈`CppImplementationDataType` of `category VECTOR` that boils down to `ara::core::Vector` shall aggregate

- one `templateArgument` that defines the type of elements contained in the `CppImplementationDataType` of `category VECTOR` with the `templateType` reference.
- optionally one additional `templateArgument` that defines the `Allocator` with the `allocator` reference.

⌋*()*

[TPS_MANI_03186]{DRAFT} **Usage of `arraySize` in case of a Vector** ⌈If the `CppImplementationDataType` of `category VECTOR` aggregates a `templateArgument` that defines the `Allocator` with the `allocator` reference then the attribute `arraySize` that defines the maximum size of the vector is allowed to be used.⌋*(RS_MANI_00039)*

Figure 3.11 shows an example of an one-dimensional vector of uint16 elements.

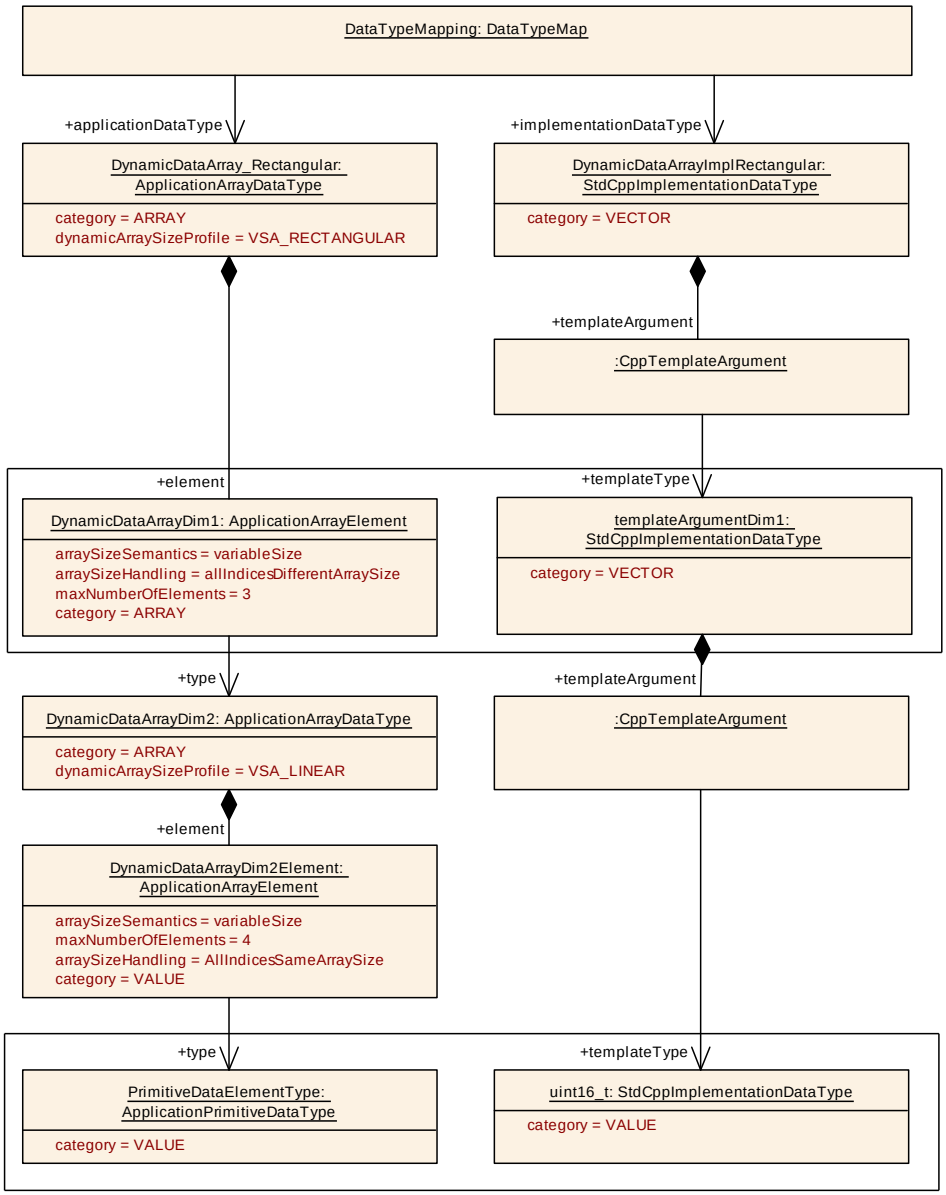


Figure 3.12: Example of the model of a multidimensional vector

[TPS_MANI_03177]{DRAFT} **Definition of a multidimensional Vector** ⌈A multidimensional `CppImplementationDataType` of `category VECTOR` contains nested `CppImplementationDataType`s of `category VECTOR`.

The `CppImplementationDataType` of `category VECTOR` that represents the outer vector will refer to a `CppImplementationDataType` of `category VECTOR` that represents the inner vector via the aggregated `templateArgument`.

Such a definition describes a two-dimensional Vector; consequently a type with more dimensions is described by just nesting more `CppImplementationDataType`s of `category VECTOR`.

The vector element itself is specified by the innermost `CppImplementationDataType` with `category` different from `VECTOR`.⌋*(RS_MANI_00039)*

Figure 3.12 shows an example of a multidimensional vector where a CppImplementationDataType of category VECTOR has a templateArgument that points to the inner CppImplementationDataType of category VECTOR in the role templateType. The inner CppImplementationDataType has a templateArgument that finally points with the templateType reference to a primitive type.

Please note that the meta-model supports the creation of a reference to a specific element (identified by means of the index) of a CppImplementationDataType of category VECTOR.

However, this may lead to a problem at run-time if the specific element does not exist at the respective point in time. Any software using such data types needs to be prepared for the potential non-existence of vector elements.

Alternatively, it could be an option to simply avoid a situation where an element of a CppImplementationDataType of category VECTOR becomes the target of a reference in the model.

3.3.3.7 Struct Data Type

[TPS_MANI_03180]{DRAFT} **Definition of Structures** ⌈A StdCppImplementationDataType of category STRUCTURE represents a data type for holding an ordered collection of variables of arbitrary data types.⌋*(RS_MANI_00039)*

[TPS_MANI_03181]{DRAFT} **Definition of members in StdCppImplementationDataType of category STRUCTURE** ⌈Members in a StdCppImplementationDataType of category STRUCTURE are defined by ordered CppImplementationDataTypeElements that are aggregated in the role subElement by the enclosing StdCppImplementationDataType of category STRUCTURE.

The name of each member is defined by the shortName of the CppImplementationDataTypeElement.

The type of each member is defined by the typeReference to a CppImplementationDataType.⌋*(RS_MANI_00039)*

Please note that the inplace flag that is able to classify a CppImplementationDataTypeElement.typeReference is documented in [TPS_MANI_03196].

The example depicted in Figure 3.13 shows the definition of a Structure, called MyStruct, that has two members. The typeReference of the subElements with the shortName ArrayElement is classified with inplace = **True**.

In case that the inplace attribute in the typeReference to the array is set to **False** the model results in a using-declaration of ArrayDataType that is defined outside MyStruct.

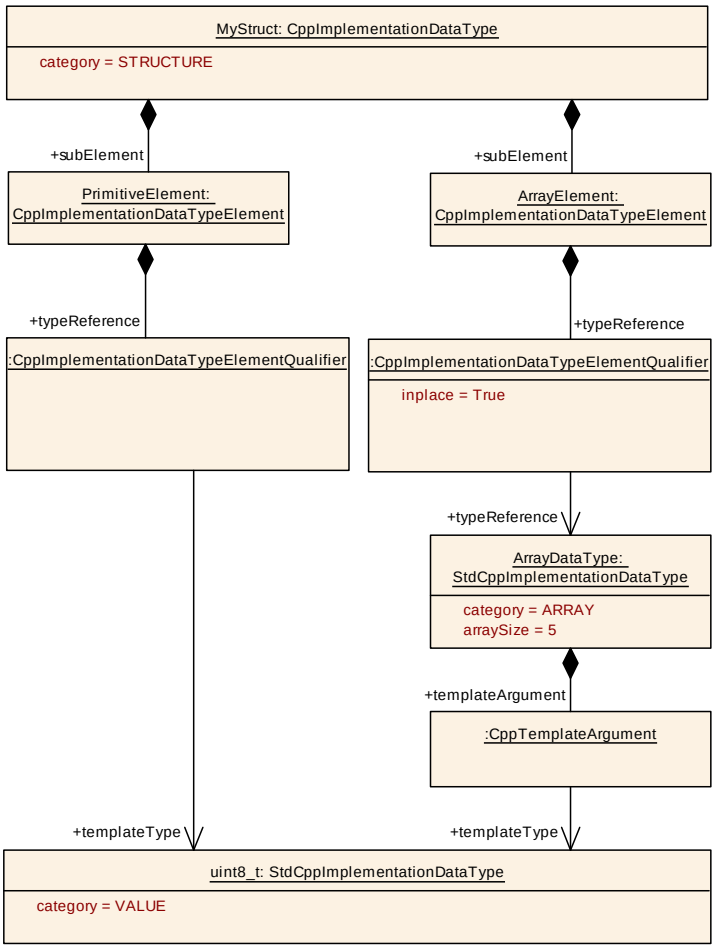


Figure 3.13: Example of the model of a Struct

3.3.3.8 Enumeration Data Type

[TPS_MANI_03187]{DRAFT} **Definition of enumeration types** ⌈In the AUTOSAR meta-model, an enumeration is not implemented by means of a `CppImplementationDataType` with an own `category`.

Instead, a discrete set of integer numbers can be used as a structural description for a single fundamental `CppImplementationDataType` of `category` `TYPE_REFERENCE` that boils down to a `CppImplementationDataType` of category `VALUE`.

The mapping of the integer numbers to labels in the scope of the definition of an enumeration is considered part of the semantical definition via an attached `CompuMethod` with `category` `TEXTTABLE` rather than part of the structural description.⌋*(RS_MANI_00039)*

The rules for the usage of a `CompuMethod` with `category` `TEXTTABLE` are the same as in the AUTOSAR Classic Platform and are described in the Software Component Template [1].

To summarize, an enumeration value in the CompuMethod with category TEXTTABLE can be provided as a text value in the vt of the CompuConst, in the shortLabel or symbol of the applicable CompuScale of the CompuMethod.

Each CompuScale shall be defined as compuInternalToPhys computation in the CompuMethod and shall contain an upperLimit and lowerLimit.

The following example illustrates how an enumeration is specified using a CompuMethod.

Listing 3.2: example for enumeration

```
<COMPU-METHOD>
  <SHORT-NAME>cylinders</SHORT-NAME>
  <CATEGORY>TEXTTABLE</CATEGORY>
  <COMPU-INTERNAL-TO-PHYS>
    <COMPU-SCALES>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>Cylinder1</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">1</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">1</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>Cylinder2</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">2</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">2</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>Cylinder3</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">3</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">3</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>Cylinder4</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
    </COMPU-SCALES>
  </COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
```

3.3.3.9 Map Data Type

[TPS_MANI_03183]{DRAFT} **CppImplementationDataType of category ASSOCIATIVE_MAP** ⌈A CppImplementationDataType of category ASSOCIATIVE_MAP represents a container that contains key-value pairs with unique keys.⌋ *(RS_MANI_00039)*

[TPS_MANI_03184]{DRAFT} **CppImplementationDataType of category ASSOCIATIVE_MAP** ⌈For a C++ binding, a CppImplementationDataType of category ASSOCIATIVE_MAP can be implemented as

- an `ara::core::Map` if StdCppImplementationDataType subclass is used or as
- a map type in a custom namespace (e.g. `my::map`) if CustomCppImplementationDataType subclass is used (provided that the type in the custom namespace can be configured with the available modeling capabilities).

⌋*(RS_MANI_00039)*

[TPS_MANI_03185]{DRAFT} **Structure of a CppImplementationDataType of category ASSOCIATIVE_MAP** ⌈A CppImplementationDataType of category ASSOCIATIVE_MAP that boils down to a `ara::core::Map` shall aggregate the following CppTemplateArguments:

- the **first** CppTemplateArgument shall refer to a CppImplementationDataType with the templateType reference.

  This CppTemplateArgument represents the role that corresponds to ApplicationAssocMapDataType.key and defines the respective data type details.

- the **second** CppTemplateArgument shall refer a CppImplementationDataType with the templateType reference.

  This CppTemplateArgument represents the role that corresponds to ApplicationAssocMapDataType.value and defines the respective data type details.

- the optional **third** CppTemplateArgument shall refer to an Allocator with the allocator reference.

⌋*(RS_MANI_00039)*

The example depicted in Figure 3.14 shows the definition of a ASSOCIATIVE_MAP that has two CppTemplateArguments, one for the key and one for the value.

Please note that the CppTemplateArguments of a CppImplementationDataType are ordered in ARXML and this order is not visible in the object diagram.

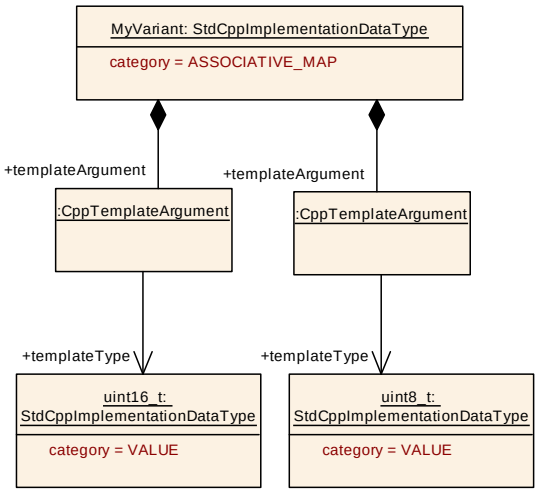


Figure 3.14: Example of the model of an ASSOCIATIVE_MAP

3.3.3.10 Variant Data Type

[TPS_MANI_03189]{DRAFT} **Definition of CppImplementationDataType of category VARIANT** ⌈A CppImplementationDataType of category VARIANT represents a type safe union.⌋*(RS_MANI_00039)*

[TPS_MANI_03190]{DRAFT} **CppImplementationDataType of category VARIANT** ⌈For a C++ binding, a CppImplementationDataType of category VARIANT can be implemented as

- an ara::core::Variant if StdCppImplementationDataType subclass is used or as
- a variant type in a custom namespace (e.g. my::variant) if CustomCppImplementationDataType subclass is used (provided that the type in the custom namespace can be configured with the available modeling capabilities).

⌋*(RS_MANI_00039)*

[TPS_MANI_03191]{DRAFT} **Definition of type alternatives stored in a VARIANT** ⌈A type alternative that is stored in a CppImplementationDataType of category VARIANT is defined by the aggregated templateArgument and the corresponding templateType that defines the data type of the CppTemplateArgument.⌋*(RS_MANI_00039)*

[constr_3429]{DRAFT} **No allocator usage for CppImplementationDataTypes of category VARIANT** ⌈CppImplementationDataType of category VARIANT is not allowed to aggregate a templateArgument that points to an Allocator in the role allocator.⌋*()*

The example depicted in Figure 3.15 shows the definition of a VARIANT that has two CppTemplateArguments. Each one represents one alternative type. Please note

that the CppTemplateArguments of a CppImplementationDataType are ordered in ARXML and this order is not visible in the object diagram.

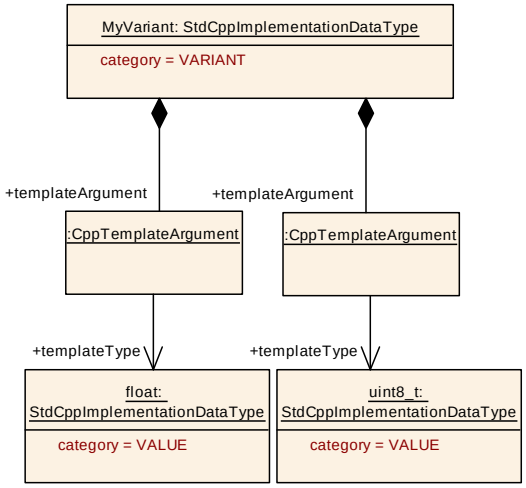


Figure 3.15: Example of the model of an VARIANT

3.3.3.11 Bitfield Data Type

[TPS_MANI_03202]{DRAFT} **Definition of bitfield types** ⌈In the AUTOSAR meta-model, a bitfield is not implemented by means of a CppImplementationDataType with an own category.

A bitfield is defined in the context of a primitive StdCppImplementationDataType of category TYPE_REFERENCE that boils down to a StdCppImplementation-DataType of category VALUE.

A CompuMethod of category BITFIELD_TEXTTABLE is used to assign a special meaning to each bit of the primitive StdCppImplementationDataType.⌋*(RS_-MANI_00039)*

CompuScales with a mask inside of the CompuMethod of category BIT-FIELD_TEXTTABLE are defining isolated parts that can be independent from each other with respect to the semantics of the data that match the mask.

The rules for the usage of a CompuMethod with category BITFIELD_TEXTTABLE are the same as in the AUTOSAR Classic Platform and are described in the Software Component Template [1].

3.3.4 Compatibility of ApplicationDataType and CppImplementationDataType

The usage of ApplicationDataTypes implies that also a corresponding CppImplementationDataType exists at a certain point in time. The usage of CppImplementationDataTypes in a ServiceInterface is required as the basis for generating the ara::com proxies and skeletons and as basis for the serialization of the payload in the network binding.

[TPS_MANI_03223]{DRAFT} **Existence of CppImplementationDataType** ⌈The existence of CppImplementationDataTypes is **not** required until the methodology step of generating the Service header files for a ServiceInterface. Before arriving at this step in the methodology, it is perfectly feasible to use only ApplicationDataTypes for describing the semantics of ServiceInterfaces.⌋*(RS_MANI_00003)*

As a consequence, it is necessary to define compatibility rules that unambiguously clarify the conformance of an ApplicationDataType with a CppImplementationDataType and vice versa.

Several rules depend on the category of the data types:

1. As a general rule, if a CppImplementationDataType of category TYPE_REFERENCE is targeted by a type mapping all the rules given below apply to the CppImplementationDataType which is finally valid after resolving all such references.

   This is not repeated in all rules. For example, if the document states that a given ApplicationDataType can be mapped to a CppImplementationDataType of category VALUE this shall include the possibility of mapping to a CppImplementationDataType of category TYPE_REFERENCE which refers to another CppImplementationDataType of category VALUE.

2. **[constr_5033]**{DRAFT} **Compatibility of data types with category VALUE** ⌈An ApplicationDataType of category VALUE can only be mapped to a CppImplementationDataType which also has category VALUE.⌋*()*

   In this case, the C++ data type resulting from the CppImplementationDataType shall be able to express all the numerical values required by the ApplicationDataType.

   This condition is fulfilled if the numerical range which can be expressed by the C++ data type at least covers the range defined by the limits in ApplicationDataType.swDataDefProps.dataConstr (which are either internal limits or physical limits to be converted via the CompuMethod which also has to be provided by the ApplicationDataType).

   The condition is also fulfilled if the C++ data type covers the range defined in the CompuMethod for an enumeration.

3. **[constr_5034]**{DRAFT} **Compatibility of data types with `category BOOLEAN`** ⌈An `ApplicationDataType` of `category BOOLEAN` can only be mapped to a `CppImplementationDataType` of `category VALUE`.⌋*()*

4. **[constr_5035]**{DRAFT} **Compatibility of data types with `category STRING`** ⌈An `ApplicationDataType` of `category STRING` can only be mapped to a `CppImplementationDataType` of `category STRING`.⌋*()*

5. **[constr_5036]**{DRAFT} **Compatibility of data types with `category ARRAY`** ⌈An `ApplicationDataType` of `category ARRAY` can only be mapped to

   - a `CppImplementationDataType` of `category ARRAY` **or**
   - a `CppImplementationDataType` of `category VECTOR`.

   ⌋*()*

   In this case, the array size and the type of the array elements of the `CppImplementationDataType` shall be such that they can be mapped/transferred 1:1 by order to the corresponding application data and vice versa.

6. **[constr_5037]**{DRAFT} **Compatibility of data types with `category ARRAY` with `variableSize`** ⌈An `ApplicationDataType` of `category ARRAY` that includes one `ApplicationArrayElement` with `arraySizeSemantics` set to `variableSize` in one of the defined dimensions shall be mapped to

   - a `CppImplementationDataType` of `category VECTOR`

   ⌋*()*

7. **[constr_5038]**{DRAFT} **Compatibility of data types with `category ARRAY` with `fixedSize`** ⌈An `ApplicationDataType` of `category ARRAY` that includes only `ApplicationArrayElement`s with `arraySizeSemantics` set to `fixedSize` in all defined dimensions shall be mapped to

   - a `CppImplementationDataType` of `category ARRAY`

   ⌋*()*

8. **[constr_5039]**{DRAFT} **Compatibility of data types with `category STRUCTURE`** ⌈An `ApplicationDataType` of `category STRUCTURE` can only be mapped to a `CppImplementationDataType` of `category STRUCTURE`.⌋*()*

   This means, that the corresponding pairs of elements shall also have compatible types.

9. **[constr_5040]**{DRAFT} **Compatibility of `ApplicationRecordDataType` and `CppImplementationDataType` that both represent an `Optional Element Structure`** ⌈An `ApplicationRecordDataType` that represents an `Optional Element Structure` can only be mapped to a `CppImplementationDataType` of `category STRUCTURE` that represents an `Optional Element Structure` if corresponding pairs of elements have the same value of the attribute `isOptional`.⌋*()*

10. **[constr_5041]**{DRAFT} **Compatibility of data types with `category` ASSOCIATIVE_MAP** ⌈An `ApplicationDataType` of `category` ASSOCIATIVE_MAP can only be mapped to a `CppImplementationDataType` of `category` ASSOCIATIVE_MAP.⌋*()*
11. **[constr_5042]**{DRAFT} **No data type mapping for `CppImplementationDataType` of `category` VARIANT** ⌈An `ApplicationDataType` shall never be mapped to a `CppImplementationDataType` of `category` VARIANT.⌋*()*
12. **[constr_5043]**{DRAFT} **Forbidden mappings to `CppImplementationDataType`** ⌈An `ApplicationDataType` of `category` COM_AXIS, RES_AXIS, CURVE, MAP, CUBOID, CUBE_4, CUBE_5 is not supported by the Adaptive Platform and can therefore not be mapped to a `CppImplementationDataType`.⌋*()*

Please note that the categories listed in [constr_5043] are not supported because there is no use case for the usage in Adaptive Platform.

On the AUTOSAR classic Platform, elements of a composite data type are not required to be considered in a `DataTypeMap`. This regulation is motivated by the fact that an element of a composite data type on the AUTOSAR classic Platform does not necessarily have a reference to an ImplementationDataType.

On the AUTOSAR adaptive Platform the situation is different. The `CppImplementationDataTypeElement` always requires a reference to a formalized `CppImplementationDataType`.

Since the processing of the data type definition becomes much easier if all the relevant data types are mentioned in a `DataTypeMap` the existence of [constr_5044] is motivated.

[constr_5044]{DRAFT} **`DataTypeMap` for composite data types** ⌈In the context of a given `ServiceInterface`, all pairs of `ApplicationDataType` and `CppImplementationDataType` used in the context of the definition of an `ApplicationCompositeDataType` used in the context of an `event`, `field`, `method` shall be described in a `DataTypeMap` that is contained in one of the `DataTypeMappingSet`s that are referenced in a `PortInterfaceToDataTypeMapping` that also references the mentioned `ServiceInterface`.⌋*()*

3.4 Service Interface

3.4.1 Overview

[TPS_MANI_01001]{DRAFT} **Meaning of `ServiceInterface`** ⌈Meta-class `ServiceInterface` inherits from `PortInterface` and allows for a heterogeneous aggregation of elements, i.e. it is possible to mix

- aggregation of `VariableDataPrototype` in the role `event` with

- aggregation of meta-class `Field` in the role `field` with
- aggregation of `ClientServerOperation` in the role `method`

within the same `ServiceInterface`.⌋*(RS_MANI_00003)*

The purpose of this modeling is to embrace the concept of service-oriented communication [3] and better support this paradigm for communication on the *AUTOSAR adaptive platform*.

Please note that, in terms of semantics, the `ApApplicationError` represents a sort of second-class citizen (that only makes sense in the presence of `ClientServerOperation` in the role `method`) in the scope of the `ServiceInterface`.

More information can be found in section 3.4.7.

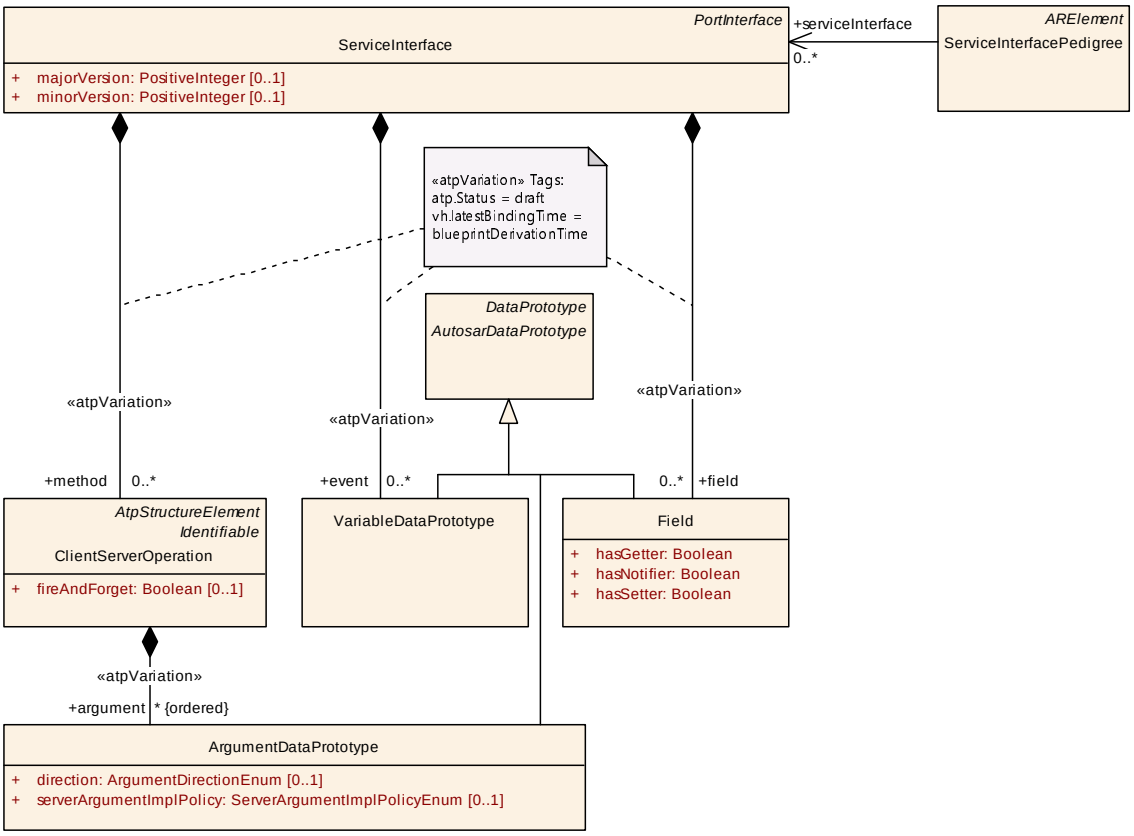


Figure 3.16: Modeling of the `ServiceInterface`

[constr_1483]{DRAFT} **Applicability of a `ServiceInterface`** ⌈The applicability of a `ServiceInterface` shall be limited to the *AUTOSAR adaptive platform*, i.e. a `ServiceInterface` shall only be taken to type a `PortPrototype` if the latter is aggregated by an `AdaptiveApplicationSwComponentType` or by a `CompositionSwComponentType` defined in the context of an `Executable`.⌋*()*

Please note that on the *AUTOSAR adaptive platform* there are use-cases for the utilization of a `ServiceInterface` **without** the existence of a corresponding `PortPrototype`. For more explanation, please refer to [TPS_MANI_01032].

Class	**ServiceInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This represents the ability to define a PortInterface that consists of a heterogeneous collection of methods, events and fields. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
event	VariableDataPrototype	*	aggr	This represents the collection of events defined in the context of a ServiceInterface. **Stereotypes:** atpVariation **Tags:** atp.Status=draft vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
field	Field	*	aggr	This represents the collection of fields defined in the context of a ServiceInterface. **Stereotypes:** atpVariation **Tags:** atp.Status=draft vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=40
majorVersion	PositiveInteger	0..1	attr	Major version of the service contract. **Tags:** atp.Status=draft xml.sequenceOffset=10
method	ClientServerOperation	*	aggr	This represents the collection of methods defined in the context of a ServiceInterface. **Stereotypes:** atpVariation **Tags:** atp.Status=draft vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=50
minorVersion	PositiveInteger	0..1	attr	Minor version of the service contract. **Tags:** atp.Status=draft xml.sequenceOffset=20

Table 3.21: ServiceInterface

[TPS_MANI_01007]{DRAFT} **Atomic unit of service discovery** ⌈As far as the application level is concerned, the atomic unit for **service discovery** on the *AUTOSAR adaptive platform* is the `ServiceInterface`.⌋*(RS_MANI_00003)*

Please note that there is no obligation to have any `method`, `event`, or `field` defined in the context of a given `ServiceInterface`. In other words, the existence of a `ServiceInterface` by itself represents a valid semantics that has a value on its own.

For example, a use case could exist where a given service instance that corresponds to such a `ServiceInterface` is offered with the mere intention to signal that the ECU that provides the service instance is becoming ready for something, e.g. being diagnosed.

A tester could then take the existence of the offer as an indication to initiate a connection to the respective ECU.

3.4.2 Event

[TPS_MANI_01033]{DRAFT} **Semantics of `ServiceInterface.event`** ⌈An `event` represents an update to a piece of data. The server decides when to send this update and makes sure that the `event` has full control over the value.

The occurrence of an `event` is transmitted from a server to one or more client(s).⌋ *(RS_MANI_00003)*

[constr_1494]{DRAFT} **Initial value for `event`** ⌈An `ServiceInterface.event` shall **not** have an `initValue`.⌋*()*

For the client, the only way to get access to the value of an `event` is to receive an update of the `event` from the server.

As mentioned in [constr_1494], the Server always has full control over the value of the `event` and when it is sent to clients. Therefore, the definition of an `initValue` is not necessary.

Class	**VariableDataPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	A VariableDataPrototype is used to contain values in an ECU application. This means that most likely a VariableDataPrototype allocates "static" memory on the ECU. In some cases optimization strategies might lead to a situation where the memory allocation can be avoided. In particular, the value of a VariableDataPrototype is likely to change as the ECU on which it is used executes.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the VariableDataPrototype

Table 3.22: VariableDataPrototype

3.4.3 Field

[TPS_MANI_01034]{DRAFT} **Semantics of `ServiceInterface.field`** ⌈A `field` represents a piece of data hosted by a server that exposes to one or more client(s) a get accessor and/or a set mutator.

Clients can optionally receive notifications of changes of the `field`'s value.⌋*(RS_MANI_00003)*

In comparison to an `event`, a `field` has a concrete value at any time. This conceptual difference can be explained along the following examples:

Let a traffic-sign detection be an example for the semantics of an event. The detection of a traffic-sign represents a discrete event in time that would be raised by the service component any time a speed limit sign is detected.

On the other hand, let a temperature preset of the in-vehicle air-condition be an example for a field that has a concrete value at any given time. The concrete value can be set by a client, can be obtained on request of a client, and – at the same time – a change of the temperature preset represents relevant information by itself.

In summary, this means that if a field is defined with hasNotifier and a client subscribes to it then the current value of the field is sent back immediately to the subscriber in an event-like notification pattern as soon as the subscription to the field becomes effective.

Additional update notifications will be sent to subscribers whenever the value of the field gets updated.

In more technical terms, the get() accessor method the current field value can be retrieved by the client. By means of calling the set() mutator method the field value can be updated by the client.

Please note that all features that a field provides are optional, given a fulfillment of [constr_1673]. In the ServiceInterface.field description it is defined whether the field supports the on-change-notification (hasNotifier), the get() accessor (hasGetter) or the set() mutator (hasSetter).

Admittedly, the concept of the field is roughly equivalent to an aggregation of an event with correlated get()/set() methods.

As far as the meta-model is concerned, the fact that a field shall have a concrete value at any time demands the **definition of an initial value** for the field. This aspect is clarified by [TPS_MANI_03212].

The existence of meta-class field as a first class citizen in the ServiceInterface expresses in addition to the existence of an individual event and individual methods that the two defined accessor/mutator methods get() and set() are applied to the **same data object** and that the defined field notifier reports each value change of this data object to subscribers.

In other words, the semantics of meta-class Field is fully determined by the attributes hasGetter, hasSetter, and hasNotifier.

Therefore, a Field where all of these attributes are set to False wouldn't have any useful meaning and shall therefore not exist.

[constr_1673]{DRAFT} **Existence of attributes hasGetter, hasSetter, and hasNotifier** ⌈For any given Field, all of the attributes

- hasGetter
- hasSetter

- hasNotifier

shall exist and at least one of the attributes shall be set to True.⌋*()*

Please note that [constr_1673] allows that a Field may be defined with a notifier but without the two defined methods get() and set(). As described above a subscriber to a field notifier will get the current value of the Field immediately after the subscription. This functionality makes a Field without get()/set() methods useful in some functional cases compared to the usage of an event where the value would only be sent after the event is triggered.

Class	**Field**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the ability to define a piece of data that can be accessed with read and/or write semantics. It is also possible to generate a notification if the value of the data changes. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
hasGetter	Boolean	1	attr	This attribute controls whether read access is foreseen to this field.
hasNotifier	Boolean	1	attr	This attribute controls whether a notification semantics is foreseen to this field.
hasSetter	Boolean	1	attr	This attribute controls whether write access is foreseen to this field.

Table 3.23: Field

3.4.4 Method

[TPS_MANI_01035]{DRAFT} **Semantics of ServiceInterface.method** ⌈A method represents a function that is executed by and in the scope of a server on request of one or more client(s).⌋*(RS_MANI_00003)*

Class	**ClientServerOperation**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	An operation declared within the scope of a client/server interface.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
argument (ordered)	ArgumentDataPrototype	*	aggr	An argument of this ClientServerOperation **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=blueprintDerivationTime
fireAndForget	Boolean	0..1	attr	This attribute defines whether this method is a fire&forget method (true) or not (false). **Tags:**atp.Status=draft

▽

△

Class	ClientServerOperation			
possibleApError	ApApplicationError	*	ref	This reference identifies AdaptivePlatformApplication Errors as a possible error raised by the enclosing Client ServerOperation. **Tags:**atp.Status=draft
possibleApError Set	ApApplicationErrorSet	*	ref	This reference represents the ability to refer to an entire group of ApApplicationErrors as one model element instead of having to refer to all the represented Ap ApplicationErrors separately. **Tags:**atp.Status=draft

Table 3.24: ClientServerOperation

3.4.4.1 Fire and Forget Method

A so-called “fire & forget” method represents a special form of a `method` dedicated to the sole purpose of conveying information from a client to a server.

There is no expectation that the implementation of the `method` executes any kind of algorithm other than to merely accept the incoming data.

Spun from this angle, the semantics of a “fire & forget” method is comparable to the semantics of an `event`, only reverse.

In other words, the “fire & forget” method conveys the data and the occurrence of the data **from a client to a server**. For comparison, the `event` is used to convey information in combination with the occurrence of the information from **a server to a client**.

The *occurrence* aspect of this statement has the consequence that e.g. the number of “fire & forget” calls can be counted by the implementation of the server and this meta-information could be taken to convey additional semantics on top of the actual data.

[TPS_MANI_01064]{DRAFT} **Semantics of attribute `method.fireAndForget`** ⌈The activation of the “fire & forget” semantics of a given `method` is achieved by setting the value of attribute `method.fireAndForget` to value `true`.⌋*(RS_MANI_00003)*

[TPS_MANI_03118]{DRAFT} **Semantics of `ServiceInterface.method` with `fireAndForget` set to true** ⌈A `method` with `fireAndForget` set to the value `true` represents a void-return-method where the client is not expecting any kind of acknowledge or handshake from the server side.⌋*(RS_MANI_00003)*

[constr_3374]{DRAFT} **`method` with attribute `fireAndForget` set to true shall not have any inout or out arguments** ⌈A `method` that has the value of attribute `fireAndForget` set to `true` is not allowed to have any `argument`s with `direction` `inout` or `out`.⌋*()*

[constr_3375]{DRAFT} **method with attribute fireAndForget set to true shall not reference an ApApplicationError** ⌈A method that has the value of attribute fireAndForget set to true is not allowed to reference

- an ApApplicationError in role possibleApError and/or
- an ApApplicationErrorSet in the role possibleApErrorSet.

⌋*()*

[TPS_MANI_03119]{DRAFT} **Default value for the attribute fireAndForget of meta-class ClientServerOperation** ⌈If the attribute fireAndForget is not defined then it shall be assumed that no "fire & forget" semantics is intended.⌋*(RS_MANI_00003)*

3.4.5 Versioning of ServiceInterfaces

Using multiple versions of the same ServiceInterface supports an independent life cycle of services and allows to change and enhance ServiceInterfaces without affection of existing consumers. This chapter describes how different versions of the same ServiceInterface can be modeled.

A version of a ServiceInterface may be defined for example as ServiceInterface with an own shortName (e.g. Service_Version1, Service_Version2) or as ServiceInterface that is located in an own ARPackage (e.g. /Version1/Service, /Version2/Service).

It is also allowed to assign a different *namespace* to the different ServiceInterface versions to influence the generated code, e.g. to generate com::version1::Service and com::version2::Service.

It is expected that if using different versions of the same ServiceInterface in one Executable then different *namespace*s shall be used for each ServiceInterface version.

The attributes ServiceInterface.majorVersion and ServiceInterface.minorVersion provide the possibility to define version information at the level of the ServiceInterface.

[TPS_MANI_03616]{DRAFT} **Semantic versioning of ServiceInterface.majorVersion and ServiceInterface.minorVersion** ⌈Service contract versioning rules:

- for backwards-incompatible interface or behavior changes the majorVersion number shall be increased and the minorVersion number shall be set to 0
- for backwards-compatible interface or behavior changes the majorVersion number shall be unchanged and the minorVersion number shall be increased.

⌋*(RS_MANI_00064)*

Note that it is expected that the decision about backwards compatibility is made by the service designer. In other words AUTOSAR does not define formal criteria for the backwards compatibility of ServiceInterfaces.

As for the modeling of several versions of a ServiceInterface, the fully qualified shortNames of the ServiceInterfaces have to be different. The ServiceInterfacePedigree allows to collect the set of ServiceInterfaces which form the collection of different versions of the same *Service*.

Class	**ServiceInterfacePedigree**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	Collection of ServiceInterfaces that belong to the same versioning. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInterfacePedigrees			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
serviceInterface	ServiceInterface	*	ref	Reference to the ServiceInterfaces which belong to the same versioning. **Tags:**atp.Status=draft

Table 3.25: ServiceInterfacePedigree

The other consumers of this service do not need to switch to using the latest version of this ServiceInterface, but can continue to use older versions of the ServiceInterface they were designed for and tested with.

3.4.5.1 Versioning driven by transport layer

Each transport layer mechanism (e.g. SOME/IP) may define its own compatibility rules. Therefore, for each individual transport layer an own impact assessment on the compatibility needs to be performed whether the changed service interface has an incompatible representation on this transport layer.

The compatibility depends on the features that are used on the transport layer. For example, in SOME/IP a length field that is put in front of a struct allows that during deserialization unknown elements at the end of an extensible data struct are skipped.

An additional option in SOME/IP is the usage of Data IDs in front of optional struct members. With this approach the receiver can skip unknown members of the struct, i.e. where the Data ID is unknown.

Therefore, on the Application Design level, all changes of ServiceInterfaces shall be handled carefully since only the used transport layer and the used features on the transport layer decide whether the change is compatible or not.

If one wants to make sure that two AutosarDataPrototypes inside a ServiceInterface are compatible then both AutosarDataPrototypes shall be typed by an identical AutosarDataType.

During the ServiceInterfaceDeployment the ServiceInterface is mapped to a middleware transport layer where the necessary middleware transport layer specific configuration settings are performed, as described in chapter 10.1.

For example, it is possible to assign the same SOME/IP serviceInterfaceId to different versions of the same ServiceInterface, but a different majorVersion or minorVersion.

This approach takes into account that the compatibility of ServiceInterfaces is heavily influenced by the used transport binding.

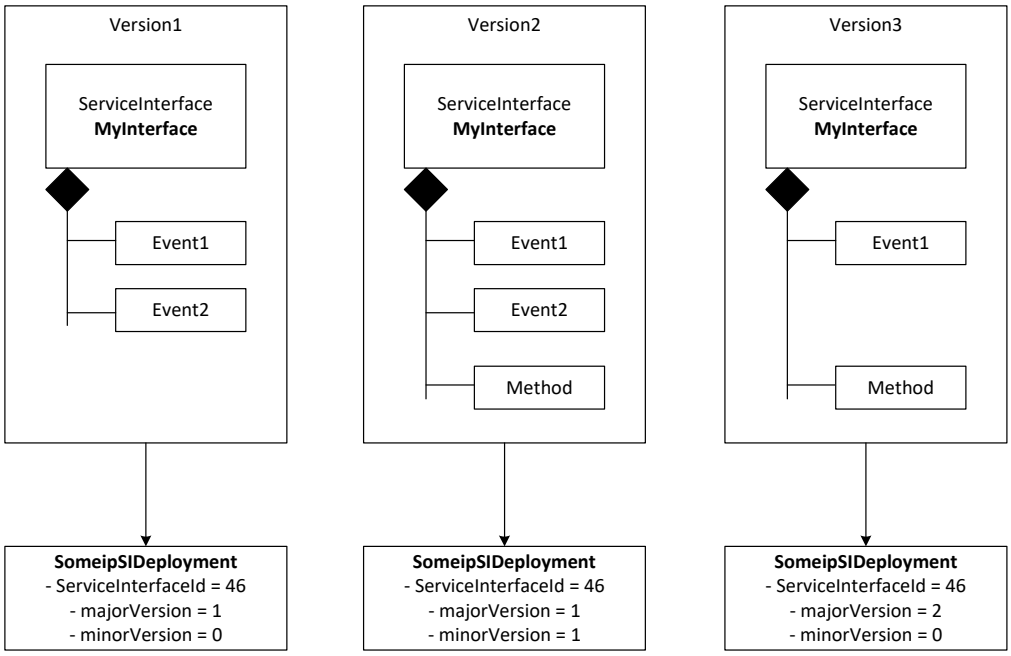


Figure 3.17: Example for different versions of the same ServiceInterface

Please note that the compatibility rules for SOME/IP are described in [7].

3.4.6 Namespace

The definition of a ServiceInterface has a direct impact on the code of an application on the *AUTOSAR adaptive platform*.

Without going into too much detail at this point, it is necessary to support the definition of a *namespace* in the context of a ServiceInterface.

The namespace shall be used to encapsulate source code related to the ServiceInterface and thus avoid name clashes with the content of other definitions of ServiceInterfaces.

In principle, the definition of the namespace around a concrete ServiceInterface could be derived from the structure of ARPackages in which the definition of the

ServiceInterface is contained. However, this approach puts some constraints of the package structure.

The same ServiceInterface may be used in different projects that may or may not demand the usage of a specific *different* package structure.

This placement of the same ServiceInterface in potentially different package hierarchies would lead to the definition of different namespaces, and thus the necessity to create or generate the code representing the ServiceInterface **plus** the code that uses this definition again and again.

One way to overcome this potential issue is to attach a dedicated namespace definition to the definition of the ServiceInterface itself.

This approach is documented in Figure 3.18.

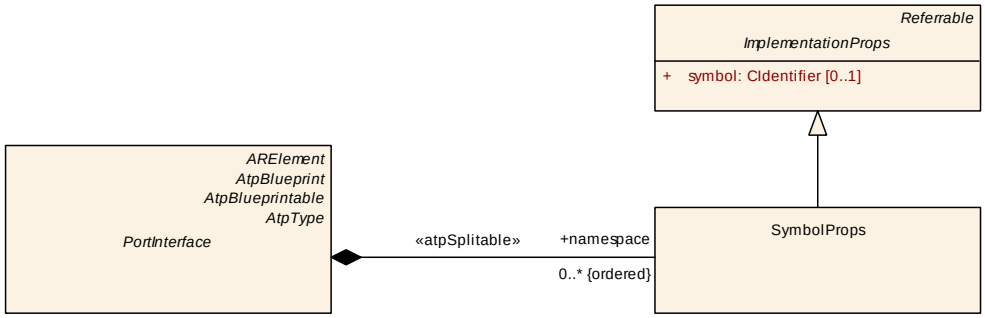


Figure 3.18: Specification of namespaces in PortInterfaces

[TPS_MANI_01004]{DRAFT} **Semantics of ServiceInterface.namespace** ⌈The aggregation ServiceInterface.namespace shall be used to define the namespace to be used for the source code that corresponds to the given ServiceInterface.⌋ *(RS_MANI_00003)*

[TPS_MANI_01005]{DRAFT} **The definition of the namespace of a ServiceInterface may follow a hierarchical pattern** ⌈The namespace of a ServiceInterface may follow a hierarchical pattern, as supported by many modern programming languages.

The separator between the elements of the hierarchical namespace definition depends on the used programming language and is not explicitly defined in the model.

The model only defines the elements of the hierarchical namespace pattern.⌋ *(RS_MANI_00003)*

As the consequence of the ability to define a hierarchical namespace, the aggregation ServiceInterface.namespace is qualified as being ordered.

This means that the order of individual elements to the collection of namespaces has a semantical relevance[4].

[4]This means that the definition of a namespace a::b is semantically different from the definition of a namespace b::a.

[TPS_MANI_01006]{DRAFT} **Ordered definition of `ServiceInterface.namespace`** ⌈In a hierarchical definition of `ServiceInterface.namespace` the order of `namespace` fragments shall be maintained in the translation of the namespace to source code.

In other words, the first `namespace` fragment shall appear first, followed by the second `namespace` fragment, and so on.⌋*(RS_MANI_00003)*

Listing 3.3: Example for the definition of a namespace for a given `ServiceInterface`

```
<SERVICE-INTERFACE>
  <SHORT-NAME>MyServiceInterface</SHORT-NAME>
  <NAMESPACES>
    <SYMBOL-PROPS>
      <SHORT-NAME>first</SHORT-NAME>
      <SYMBOL>com</SYMBOL>
    </SYMBOL-PROPS>
    <SYMBOL-PROPS>
      <SHORT-NAME>second</SHORT-NAME>
      <SYMBOL>myCompany</SYMBOL>
    </SYMBOL-PROPS>
    <SYMBOL-PROPS>
      <SHORT-NAME>third</SHORT-NAME>
      <SYMBOL>software</SYMBOL>
    </SYMBOL-PROPS>
  </NAMESPACES>
</SERVICE-INTERFACE>
```

Class	***PortInterface*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Abstract base class for an interface that is either provided or required by a port of a software component.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractRawDataStreamInterface*, *AbstractSynchronizedTimeBaseInterface*, ClientServerInterface, *CryptoInterface*, *DataInterface*, *DiagnosticPortInterface*, ModeSwitchInterface, *PersistencyInterface*, *PlatformHealthManagementInterface*, RestServiceInterface, SecurityEventReportInterface, ServiceInterface, TriggerInterface			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
namespace (ordered)	SymbolProps	*	aggr	This represents the SymbolProps used for the definition of a hierarchical namespace applicable for the generation of code artifacts out of the definition of a ServiceInterface. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=namespace.shortName atp.Status=draft

Table 3.26: PortInterface

Class	SymbolProps			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This meta-class represents the ability to contribute a part of a namespace.			
Base	*ARObject*, *ImplementationProps*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.27: SymbolProps

The Listing 3.3 exemplifies the statement made by [TPS_MANI_01006], i.e. the resulting name space in e.g. C++ would look like sketched in Listing 3.4.

```
namespace com {
    namespace myCompany {
        namespace software {

        }
    }
}
```

Listing 3.4: Resulting namespace for the example `ServiceInterface`

3.4.7 Error Handling

The modeling of error handling on the *AUTOSAR adaptive platform* slightly differs from the approach implemented on the *AUTOSAR classic platform*.

In particular, the formal representation of an error during the execution of a `method` is done in a global scope, i.e. such a definition can be reused arbitrarily by any `ServiceInterface`.

[TPS_MANI_01190]{DRAFT} **Semantics of `ApApplicationError`** ⌈Meta-class `ApApplicationError` represents the ability to define the existence of an error during the execution of a `method` independently of the scope of a `ServiceInterface` or `ClientServerOperation`.⌋*()*

Class	ApApplicationError			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the ability to formally specify the semantics of an application error on the AUTOSAR adaptive platform **Tags:** atp.Status=draft atp.recommendedPackage=ApplicationErrors			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**ApApplicationError**			
errorCode	Integer	1	attr	This attribute has the ability to specify the error code value within the enclosing AdaptivePlatformApplication Error.
errorDomain	ApApplicationError Domain	1	ref	This reference represents the error domain of the Ap ApplicationError. **Tags:**atp.Status=draft

Table 3.28: ApApplicationError

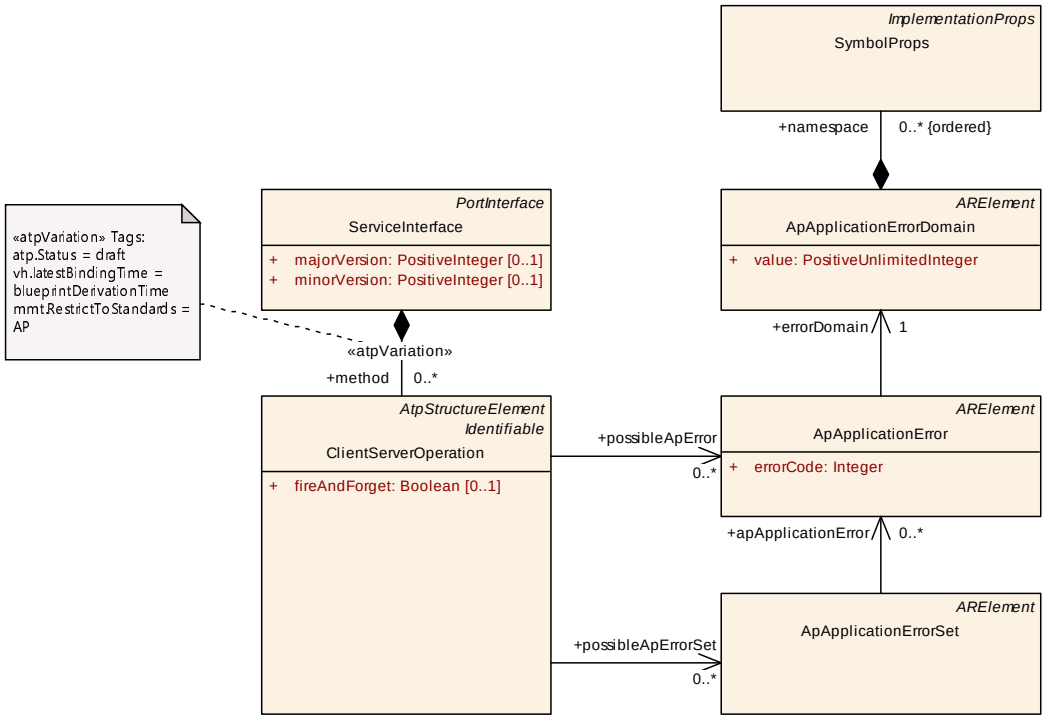


Figure 3.19: Modeling of `ApApplicationError` on the *AUTOSAR adaptive platform*

[TPS_MANI_01198]{DRAFT} **Semantics of `ApApplicationErrorSet`** ⌈Meta-class `ApApplicationErrorSet` has the ability to group references to `ApApplicationError` and thus represents a “proxy” to this group of references towards the `ClientServerOperation`.

The use case for this modeling ability is that some `ClientServerOperation`s may have to reference an identical significant number of `ApApplicationError`s.

Letting each of the `ClientServerOperation`s repeat the same set of references to `ApApplicationError` is considered unnecessary and therefore the ability to refer to a group instead of individual references is provided as an alternative.⌋*(RS_MANI_-00026)*

The decision whether an `ApApplicationErrorSet` is defined and referenced from specific `ClientServerOperation`s has to be done on an individual basis. AUTOSAR just wants to make this business as straightforward as possible.

Please note that it is also positively possible to mix the usage of ClientServerOperation.possibleApError and ClientServerOperation.possibleApErrorSet.

Class	**ApApplicationErrorSet**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class acts as a reference target that represents an entire collection of APApplicationErrors. This takes the burden from ClientServerOperations that reference a larger number of ApApplication Errors. **Tags:** atp.Status=draft atp.recommendedPackage=ApplicationErrorSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
apApplication Error	ApApplicationError	*	ref	Thi reference represents the collection of ApApplication Error represented by the enclosing ApApplicationErrorSet **Tags:**atp.Status=draft

Table 3.29: ApApplicationErrorSet

As ApApplicationError is no longer defined within the scope of a ServiceInterface, there is no need to define a mapping between two ApApplicationErrors by means of a dedicated sub-class of ServiceInterfaceElementMapping.

[TPS_MANI_01191]{DRAFT} **Modeling of possible errors** ⌈A ClientServerOperation aggregated by a ServiceInterface in the role method shall reference

- one or more ApApplicationError(s) in the role possibleApError
- one or more ApApplicationErrorSet(s) in the role possibleApErrorSet

to formally specify the existence of possible errors raised by the ClientServerOperation.⌋*(RS_MANI_00026)*

[TPS_MANI_01192]{DRAFT} **Semantics of ApApplicationErrorDomain** ⌈Meta-class ApApplicationErrorDomain shall be used to define a specific error domain that can potentially be standardized by AUTOSAR.

Therefore, the definition of such an error domain is not defined in the scope of the ApApplicationError itself. Instead, an ApApplicationError identifies the applicable error domain by means of a reference in the role errorDomain.

It is possible to attach the definition of a namespace to ApApplicationErrorDomain because this information is relevant for the language binding.⌋*(RS_MANI_00026)*

[constr_1627]{DRAFT} **Supported value range for attribute ApApplicationErrorDomain.value** ⌈The supported value range of attribute ApApplicationErrorDomain.value is limited to the interval [0..18446744073709551616].⌋*()*

Class	**ApApplicationErrorDomain**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the ability to define a global error domain for an ApApplicationError. **Tags:** atp.Status=draft atp.recommendedPackage=ApplicationErrorDomains			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
namespace (ordered)	SymbolProps	*	aggr	This aggregation defines the namespace of the Ap ApplicationErrorDomain **Tags:**atp.Status=draft
value	PositiveUnlimitedInteger	1	attr	This attribute identifies the error category.

Table 3.30: ApApplicationErrorDomain

[constr_1625]{DRAFT} **Existence of reference `ApApplicationError.errorDomain`** ⌈For each `ApApplicationError`, the reference `errorDomain` shall exist.

In other words, the association of an `ApApplicationError` with a corresponding `ApApplicationErrorDomain` is mandatory.⌋*()*

[constr_1664]{DRAFT} **Unique `ApApplicationError.shortName`** ⌈Within the set of all `ApApplicationError`s that reference a given `ApApplicationErrorDomain` in the role `errorDomain` the attribute `ApApplicationError.shortName` shall have a unique value.⌋*()*

[constr_1665]{DRAFT} **Unique `ApApplicationError.errorCode`** ⌈Within the set of all `ApApplicationError`s that reference a given `ApApplicationErrorDomain` in the role `errorDomain` the attribute `ApApplicationError.errorCode` shall have a unique value.⌋*()*

Rationale for the existence of [constr_1664] and [constr_1665]: the language binding for C++ foresees the usage of attributes `ApApplicationError.shortName` and `ApApplicationError.errorCode` for the creation of an `enum` within the context of the `ApApplicationErrorDomain`.

Duplicates in terms of labels of enumerators or values of enumerators lead to compile-time errors.

3.4.8 Service Interface Data Type Mapping

An important step in the workflow of implementing software on the *AUTOSAR adaptive platform* is the creation of a code-based representation of a `ServiceInterface` to make it accessible for the application code.

This creation of a code-based representation is usually automatized and will be executed by a code generator. This code generator needs an input from the model. The main input for this purpose is obviously the definition of the `ServiceInterface` itself.

However, this is not sufficient. The designer of a ServiceInterface is free to use ApplicationDataTypes for the specification of the details of the ServiceInterface.

It is therefore necessary to provide the definition of an AbstractImplementationDataType for each of the used ApplicationDataType. In the meta-model, this correspondence is implemented by means of the meta-class DataTypeMappingSet[5].

However, from the methodological point of view it is considered inappropriate to let ServiceInterface directly refer to one or more DataTypeMappingSet(s).

For clarification, this would mean that the mapping of ApplicationDataType to AbstractImplementationDataType becomes an integral part of the definition of the ServiceInterface although the mapping itself does not really contribute to the actual semantics of the ServiceInterface.

As a consequence, the ServiceInterface would have to be updated whenever the mapping between data types changes.

But since the definition of ServiceInterfaces are usually considered very stable a frequent update for the mere purpose of acknowledging a change in the data type mapping is not acceptable.

In this concrete case, the described problem can be circumvented by the definition of a mapping class that refers to both a ServiceInterface and a DataTypeMappingSet and therefore create the correspondence without the need to update the ServiceInterface.

Although the prelude into this chapter suggests the existence of a meta-class that maps a ServiceInterface to one or more DataTypeMappingSet(s) the actual meta-model is designed with a broader focus.

In the future, there could be further kinds of PortInterfaces beside the ServiceInterface that need to fulfill the same use case.

Consequently, the name of the meta-class created for this purpose is PortInterfaceToDataTypeMapping.

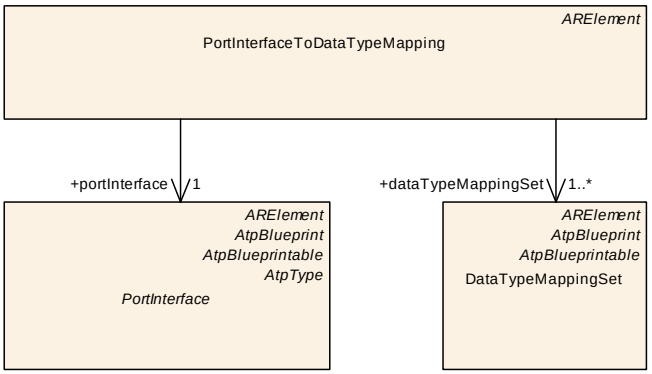


Figure 3.20: Modeling of PortInterfaceToDataTypeMapping

[5]For more background regarding the definition and use of meta-class DataTypeMappingSet please refer to [1].

[constr_1507]{DRAFT} **PortInterfaceToDataTypeMapping is only applicable to ServiceInterface or PersistencyKeyValueStorageInterface** ⌈PortInterfaceToDataTypeMapping.portInterface shall only refer to **either** a ServiceInterface **or** a PersistencyKeyValueStorageInterface.⌋()

Class	**PortInterfaceToDataTypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the ability to associate a PortInterface with a DataTypeMappingSet. This association is needed for the generation of header files in the scope of a single PortInterface. The association is intentionally made outside the scope of the PortInterface itself because the designers of a PortInterface most likely will not want to add details about the level of ImplementationDataType. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInterfaceToDataTypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataType MappingSet	DataTypeMappingSet	1..*	ref	This represents the reference to the applicable data TypemappingSet **Tags:** atp.Status=draft atp.StatusComment=Reserved for adaptive platform
portInterface	PortInterface	1	ref	This represents the reference to the applicable Port Interface **Tags:** atp.Status=draft atp.StatusComment=Reserved for adaptive platform

Table 3.31: PortInterfaceToDataTypeMapping

Class	**DataTypeMappingSet**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	This class represents a list of mappings between ApplicationDataTypes and ImplementationDataTypes. In addition, it can contain mappings between ImplementationDataTypes and ModeDeclarationGroups. **Tags:**atp.recommendedPackage=DataTypeMappingSets			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataTypeMap	DataTypeMap	*	aggr	This is one particular association between an Application DataType and its AbstractImplementationDataType.
modeRequest TypeMap	ModeRequestTypeMap	*	aggr	This is one particular association between an Mode DeclarationGroup and its AbstractImplementationData Type.

Table 3.32: DataTypeMappingSet

Class	**DataTypeMap**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	This class represents the relationship between ApplicationDataType and its implementing Abstract ImplementationDataType.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
applicationData Type	ApplicationDataType	0..1	ref	This is the corresponding ApplicationDataType
implementation DataType	AbstractImplementation DataType	0..1	ref	This is the corresponding AbstractImplementationData Type.

Table 3.33: DataTypeMap

3.4.9 Communication Group pattern

The Communication Group defines a specific pattern of the usage of a ServiceInterface in a bi-directional way.

The details can be found in SWS_CommunicationManagement [8]. In order to define a Communication Group several ServiceInterface.category values are defined.

[TPS_MANI_03628]{DRAFT} **Standardized values of ServiceInterface.category** ⌈The AUTOSAR Standard reserves the following values for attribute ServiceInterface.category:

- COMMUNICATION_GROUP
- COMMUNICATION_GROUP_SERVER
- COMMUNICATION_GROUP_CLIENT

It is possible to use a custom, non-standardized value for the attribute ServiceInterface.category but this option comes with the obligation to use a value that is guaranteed to not clash with possible future extensions of the collection of standardized values, e.g. use company name in the category value.⌋*()*

The general idea of the Communication Group pattern is that a ServiceInterface of category COMMUNICATION_GROUP is created to describe the information to be transported (the msg and responseMsg data types). There will not be any instance of this ServiceInterface of category COMMUNICATION_GROUP in the system, it is just a design artifact.

Out of the ServiceInterface of category COMMUNICATION_GROUP the two ServiceInterfaces for the server (category COMMUNICATION_GROUP_SERVER) and the client (category COMMUNICATION_GROUP_CLIENT) roles are created. The rules how this creation shall be done are defined in SWS_CommunicationManagement [8].

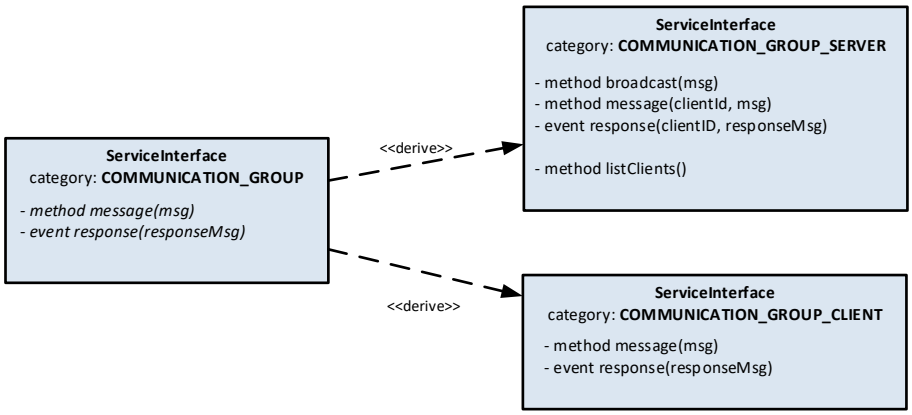


Figure 3.21: Example of Communication Group categories

3.5 Service Interface Mapping

Please note that, according to [TPS_MANI_01007], the `ServiceInterface` becomes the single basis for both `VFB`-based and *external* (i.e. using communication networks) communication.

This concept is in stark contrast to the approach on the *AUTOSAR classic platform* where different model elements are used for the `VFB`-level (`PortInterface`) and the network-level (`SystemSignal`, `ISignal`, and `ISignalIPdu`).

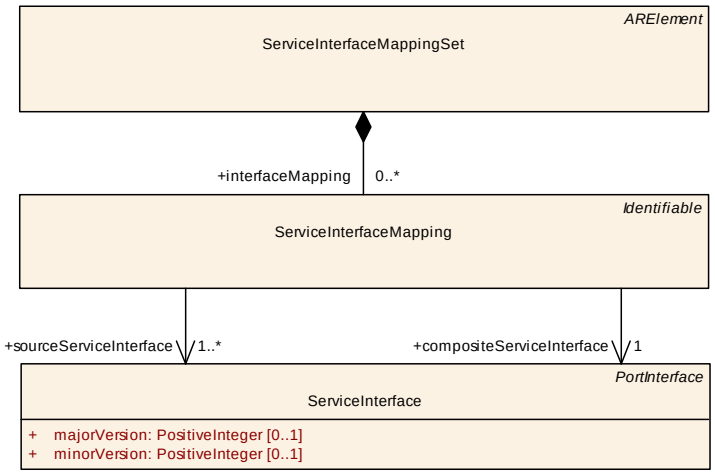


Figure 3.22: Modeling of the `ServiceInterfaceMapping`

The usage of different model elements optimally supports the existence of different granularity for `VFB`-based vs. network-based communication.

In other words, design of communication on the network level may be subject to different design restrictions, e.g. keep the bus load caused by service discovery manageable by defining coarse-grained communication packages.

Opposed to that, designers on the VFB level may want to define interface granularity to achieve maximum reusability.

[TPS_MANI_01002]{DRAFT} **Semantics of meta-class ServiceInterfaceMapping** ⌈In order to sort out a potentially different motivation between the definition of

- ServiceInterfaces explicitly designed for VFB-based communication and
- ServiceInterfaces explicitly designed for network-based communication

meta-class ServiceInterfaceMapping is available to map

- (fine-grained) ServiceInterfaces for the VFB-communication to
- (coarse-grained) ServiceInterfaces for network communication.

⌋*(RS_MANI_00017)*

[TPS_MANI_01032]{DRAFT} **Usage of ServiceInterfaceMapping** ⌈It is possible to derive a dedicated AdaptiveApplicationSwComponentType that implements the mapping functionality. A SwComponentPrototype derived from this so-called *facade* software-component would expose PortPrototypes for each of the ServiceInterfaces.

Other SwComponentPrototypes could then "connect" to the PortPrototypes typed by ServiceInterfaces referenced in the role sourceServiceInterface.

This means that the PortPrototype typed by the ServiceInterface referenced in the role compositeServiceInterface is used for external communication.

PassThroughSwConnectors can be used to describe in the modeled *facade* CompositionSwComponentType which "fine-grained" Ports are combined to a "coarse-grained" Port that is used for network communication. The mapping of Service Interface elements of the "fine-grained" Ports to the Service Interface elements of the "course-grained" Port is described with the ServiceInterfaceMapping or rather ServiceInterfaceElementMapping.⌋*(RS_MANI_00017)*

Please note that the modeling of a *facade* SwComponentType does not make any assumptions about the implementation and about the realization of such a *facade* functionality. The *facade* may be realized by an Adaptive Software Component/Application or it may be realized by a "Network-Daemon". AUTOSAR does not define any instructions for the implementation of such a functionality and the decision is project specific. The behavioral aspects of such a "facade" (e.g. when is the coarse-grained ServiceInstance offered) are also project-specific and are not predefined by AUTOSAR.

Figure 3.23 summarizes the idea behind the creation of a *facade* software-component. The latter is able to "bundle" the communication of different PortPrototypes owned by potentially different SwComponentTypes for external communication.

In other words, elements `event1` owned by `SWC1` and `event2` owned by `SWC1` are combined into one `ServiceInterface` used to type one `PortPrototype` of the *facade* software-component.

From the communication-related outside point-of-view, `SWC3` acts like a facade to the "inner structure" created by `SWC1` and `SWC2` that is, by way of the existence of `SWC3`, abstracted away.

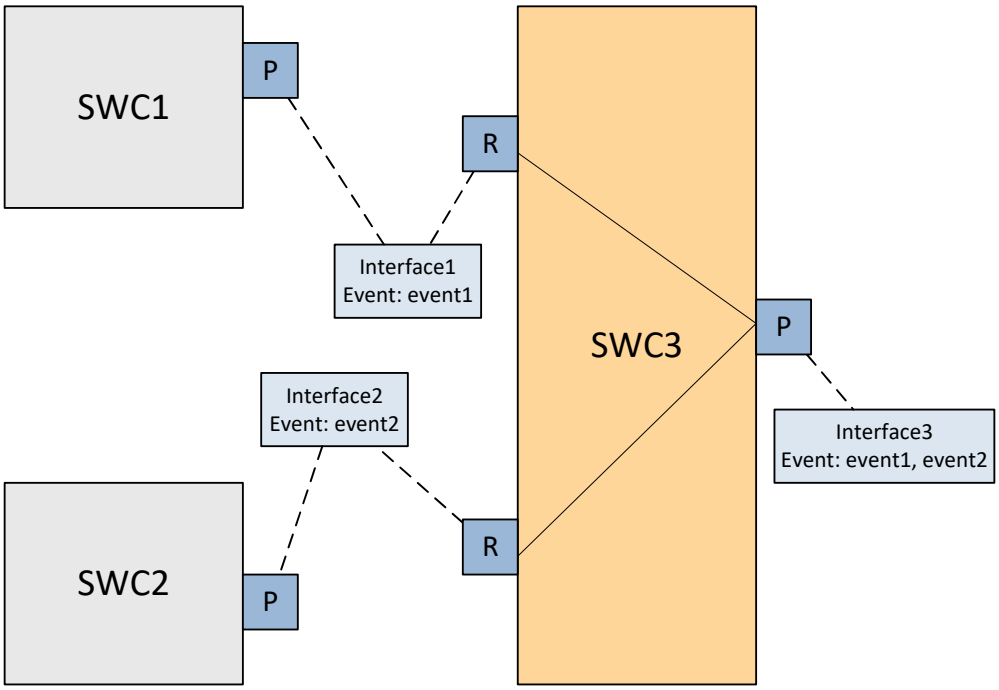


Figure 3.23: Concept of a facade software-component

[constr_5056]{DRAFT} **Restriction of `CompositionSwComponentType.connector` usage in AP** ⌈The usage of `CompositionSwComponentType.connector` on the AUTOSAR adaptive platform is restricted to `PassThroughSwConnector`s.⌋*()*

[constr_5057]{DRAFT} **`PassThroughSwConnector` and `ServiceInterfaceMapping`** ⌈If a `PassThroughSwConnector` is defined between two Ports in a `CompositionSwComponentType` then a `ServiceInterfaceMapping` or a `ServiceInterfaceElementMapping` between the `ServiceInterface`s of these two Ports shall be defined as well.⌋*()*

[TPS_MANI_01022]{DRAFT} **Concept behind `ServiceInterfaceMapping`** ⌈The concept behind the definition of a `ServiceInterfaceMapping` is that **all elements** of the `sourceServiceInterface` are required to have a **counterpart of the same kind** (`ServiceInterface.event`, `ServiceInterface.field`, or `ServiceInterface.method`) and with the identical `shortName`.⌋*(RS_MANI_00017)*

The regulation stated in [TPS_MANI_01022] is exemplified in Figure 3.24.

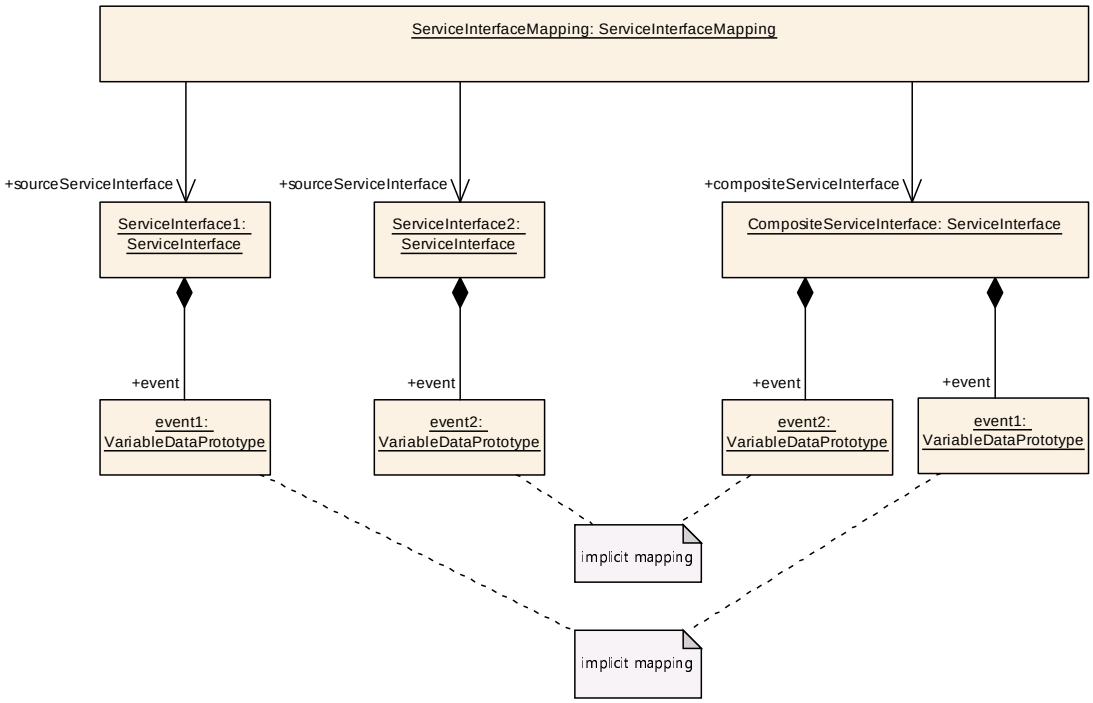


Figure 3.24: Example for the application of a `ServiceInterfaceMapping`

Please note that the creation of a `ServiceInterfaceMapping` is considered an atomic step, it is unlikely that such a `ServiceInterfaceMapping` is partially created and then later finished by a different party.

After all, there are mutually exclusive ways to specify the mapping, and any creator of a partial mapping of `ServiceInterface`s could not be sure which of the alternatives apply for a specific pairing of one `ServiceInterface` with another without already knowing the other `ServiceInterface` (in which case the mapping can already be completed).

Therefore, there is no need to set the lower multiplicity of the references to `ServiceInterface` to 0.

Class	**ServiceInterfaceMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping			
Note	Specifies one ServiceInterfaceMapping that allows to define that a ServiceInterface is composite of several other ServiceInterfaces. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compositeServiceInterface	ServiceInterface	1	ref	This represents the composite ServiceInterface. **Tags:**atp.Status=draft

△

Class	ServiceInterfaceMapping			
sourceService Interface	ServiceInterface	1..*	ref	ServiceInterface that is mapped into the composite ServiceInterface. **Tags:**atp.Status=draft

Table 3.34: ServiceInterfaceMapping

Class	ServiceInterfaceMappingSet			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping			
Note	This meta-class represents the ability to aggregate a collection of ServiceInterfaceElementMappings. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInterfaceMappingSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
element Mapping	ServiceInterface ElementMapping	*	aggr	This represents the collection of ServiceInterfaceElement Mappings aggregated at the ServiceInterfaceElement MappingSet **Tags:**atp.Status=draft
interface Mapping	ServiceInterface Mapping	*	aggr	This represents the collection of ServiceInterface Mappings owned by the ServiceInterfaceMappingSet. **Tags:**atp.Status=draft

Table 3.35: ServiceInterfaceMappingSet

[TPS_MANI_01003]{DRAFT} **Limitation of the applicability of `ServiceInterfaceMapping`** ⌈The applicability of the `ServiceInterfaceMapping` is limited to cases where the `shortName`s of the elements of the `compositeServiceInterface` are **unique** in the context of the `compositeServiceInterface`.⌋*(RS_MANI_00017)*

As already indicated, the meta-class `ServiceInterfaceMappingSet` has been defined as a container for both `ServiceInterfaceMapping`s and the `ServiceInterfaceElementMapping` introduced in section 3.6.

Note that the `ServiceInterfaceMapping` is not an up-front association (by means of `SwConnector`s) between communication ends in the sense of section 3.4.5.

As stated in [TPS_MANI_01032], the `ServiceInterfaceMapping` allows for the derivation of a facade software-component or a proper configuration of the communication middleware.

The compatibility between the `sourceServiceInterface`s and the `compositeServiceInterface` is achieved by an adequate transformation implemented in the facade software-component or the configuration of the middleware.

Thus, connecting `ServiceInterface`s (or parts of them) via `ServiceInterfaceMapping`s is not constrained by any compatibility rules apart from the ones stated in [TPS_MANI_01022].

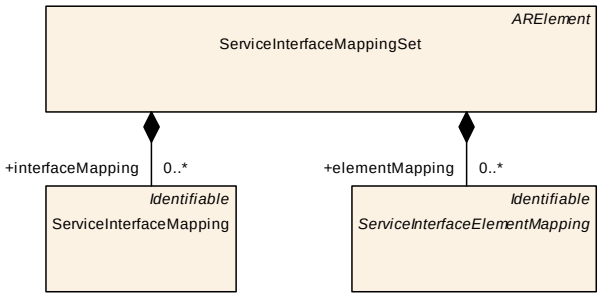


Figure 3.25: Modeling of the `ServiceInterfaceMappingSet`

3.6 Service Interface Element Mapping

3.6.1 Overview

The existence of the `ServiceInterfaceMapping` leaves the question about how `ServiceInterface`s where elements have non-matching `shortName` can be mapped.

The answer to this question is provided by the ability to create an element-wise mapping of elements of the same kind.

Figure 3.26 provides an example of how such a mapping on element basis looks like. Note that, in this example, both `ServiceInterface1` and `ServiceInterface2` aggregate a `field` with the `shortName` `field1`.

This configuration disqualifies the scenario from the application of the `ServiceInterfaceMapping`, as of [TPS_MANI_01003]. The element-wise mapping, however, is able to work around the existence of the `shortName` `field1` in both “source” `ServiceInterface`s quite nicely:

- `ServiceInterface1.field1` is mapped to `CompositeServiceInterface.leftField`
- `ServiceInterface2.field1` is mapped to `CompositeServiceInterface.rightField`

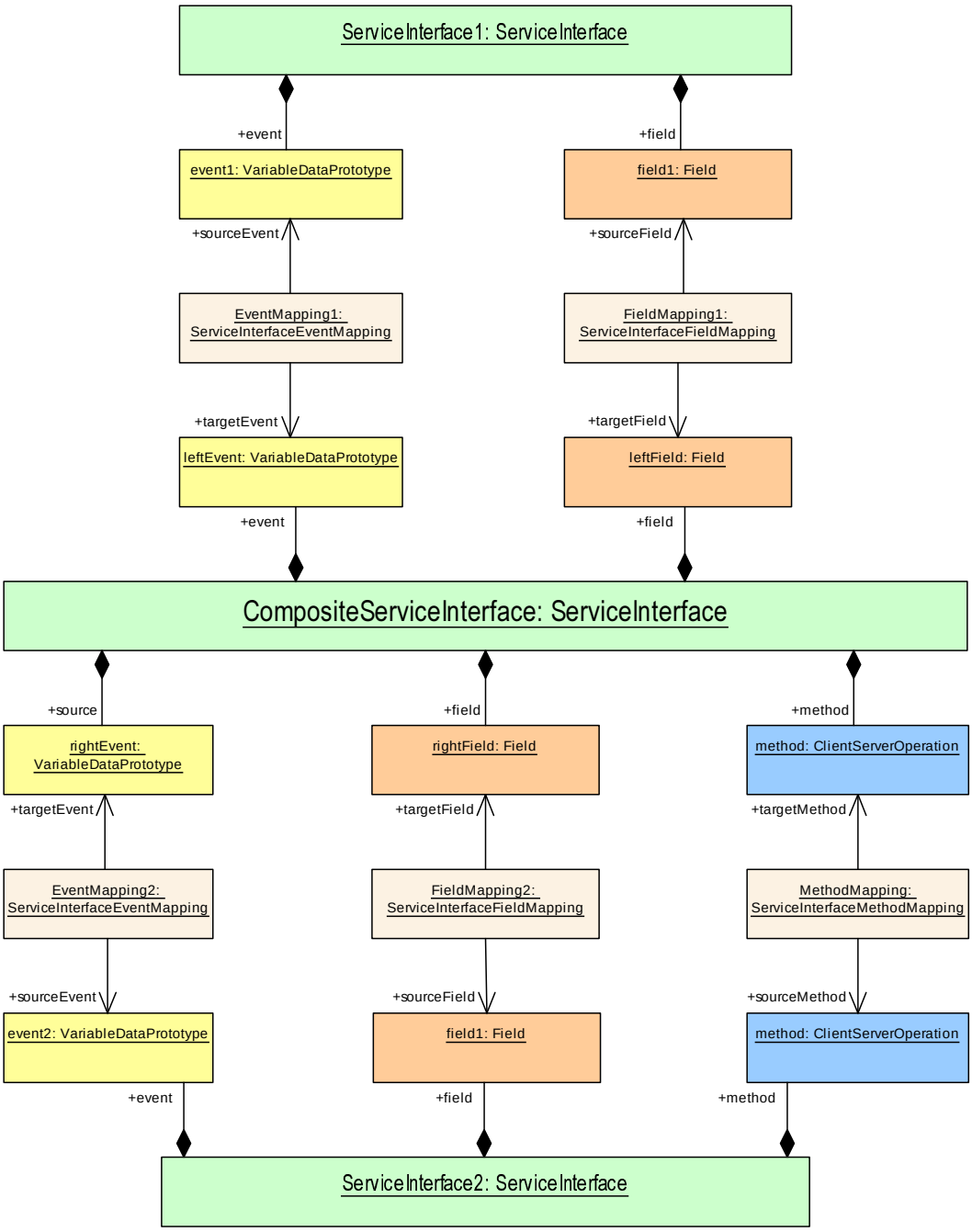


Figure 3.26: Example for a mapping of elements of `ServiceInterface`

The formal modeling of the individual mappings is described in section 3.6.

Please note that it is **not intended** to mix a mapping of `ServiceInterface`s with a mapping of elements of a `ServiceInterface`.

In other words, as soon as a mapping between two `ServiceInterface`s exists, it is not supported that a mapping between elements of the same pair of `ServiceInterface`s exists. This important restriction is formalized by [constr_1482].

[constr_1482]{DRAFT} **Mapping of service interfaces vs. mapping of service interface elements** ⌈In order to establish a mapping between a given pair of ServiceInterfaces, at most **one of** the following alternatives can exist:

- the given pair of ServiceInterfaces is referenced by a ServiceInterfaceMapping, where one ServiceInterface is referenced in the role sourceServiceInterface and the other ServiceInterface is referenced in the role compositeServiceInterface.
- an arbitrary mixture of the following options exists:
  - an event aggregated by one of the given ServiceInterfaces is referenced by a ServiceInterfaceEventMapping in the role sourceEvent and one events aggregated by the other given ServiceInterface is referenced by the same ServiceInterfaceEventMapping in the role targetEvent.
  - a field aggregated by one of the given ServiceInterfaces is referenced by a ServiceInterfaceFieldMapping in the role sourceField and one fields aggregated by the other given ServiceInterface is referenced by the same ServiceInterfaceFieldMapping in the role targetField.
  - a method aggregated by one of the given ServiceInterfaces is referenced by a ServiceInterfaceMethodMapping in the role sourceMethod and one methods aggregated by the other given ServiceInterface is referenced by the same ServiceInterfaceMethodMapping in the role targetMethod.

⌋()

Of course, it is possible that the same ServiceInterface is referenced by mappings to elements and mappings to entire ServiceInterfaces. The limitation formalized in [constr_1482] always applies to a **pair** of ServiceInterfaces.

A mapping between elements of ServiceInterfaces is modeled by means of a subclass of the abstract meta-class ServiceInterfaceElementMapping.

Class	***ServiceInterfaceElementMapping*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping			
Note	This abstract meta-class acts as base class for the mapping of specific elements of a ServiceInterface. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ServiceInterfaceEventMapping, ServiceInterfaceFieldMapping, ServiceInterfaceMethodMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.36: ServiceInterfaceElementMapping

ServiceInterfaceElementMappings are aggregated by a ServiceInterfaceMappingSet that – in principle – allows for an arbitrary grouping of ServiceInterfaceElementMappings.

Please note that the creation of a ServiceInterfaceElementMapping is considered an atomic step, i.e. it is unlikely that such a ServiceInterfaceElementMapping is partially created, handed over to a different party and then later finished by that different party.

After all, there are mutually exclusive ways to specify the mapping, and any creator of a partial mapping of ServiceInterfaces could not be sure which of the alternatives apply for a specific pairing of one ServiceInterface with another without already knowing the other ServiceInterface (in which case the mapping can already be completed).

Therefore, there is no need to set the lower multiplicity of the references to elements of the ServiceInterface to 0.

3.6.2 Service Interface Event Mapping

[TPS_MANI_01024]{DRAFT} **Semantics of ServiceInterfaceEventMapping** ⌈Meta-class ServiceInterfaceEventMapping has the ability to map a ServiceInterface.event referenced in the role sourceEvent explicitly to another ServiceInterface.event referenced in the role targetEvent.⌋*(RS_MANI_00017)*

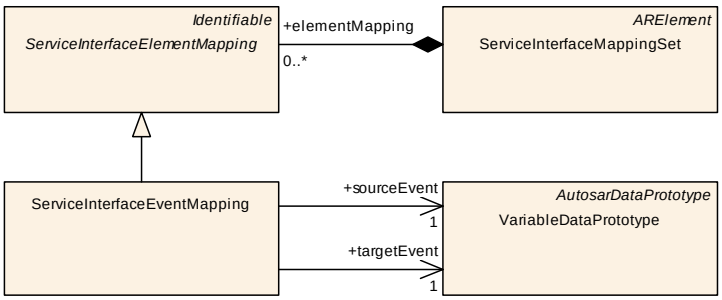


Figure 3.27: Modeling of the ServiceInterfaceEventMapping

Class	**ServiceInterfaceEventMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping			
Note	This meta-class allows to define a mapping between events of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceInterfaceElementMapping*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**ServiceInterfaceEventMapping**			
sourceEvent	VariableDataPrototype	1	ref	Reference to an event that is contained in the source ServiceInterface. **Tags:**atp.Status=draft
targetEvent	VariableDataPrototype	1	ref	Reference to an event that is contained in the composite ServiceInterface. **Tags:**atp.Status=draft

Table 3.37: ServiceInterfaceEventMapping

The explicit mapping implemented by `ServiceInterfaceEventMapping` does **not** require equal `shortName`s on both sides of the mapping.

It is also possible to map a given `event` of a given `ServiceInterface` multiple times in different roles to the `ServiceInterface` that aggregates the `targetEvent`, as exemplified by Figure 3.28.

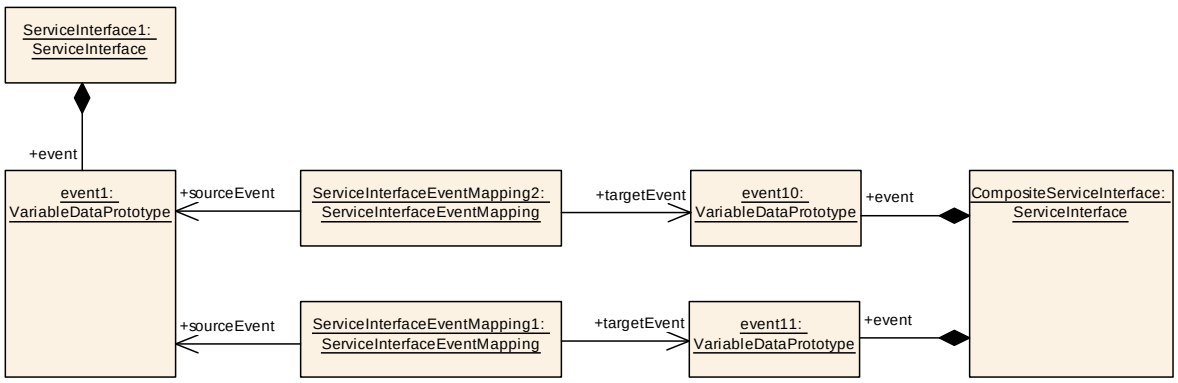


Figure 3.28: Example for the application of a `ServiceInterfaceEventMapping`

Please note that the mapping of one `sourceEvent` to different `targetEvent`s does **not** represent a *fan-out* of any kind.

It only means that the `sourceEvent` will be used in different roles, as specified in the deployment. For more explanation, please find an example of how the role-based mapping of elements of `ServiceInterface`s works in Figure A.5.

3.6.3 Service Interface Field Mapping

[TPS_MANI_01025]{DRAFT} **Semantics of `ServiceInterfaceFieldMapping`** ⌈Meta-class `ServiceInterfaceFieldMapping` has the ability to map a `ServiceInterface.field` referenced in the role `sourceField` explicitly to another `ServiceInterface.field` referenced in the role `targetField`.⌋*(RS_MANI_00017)*

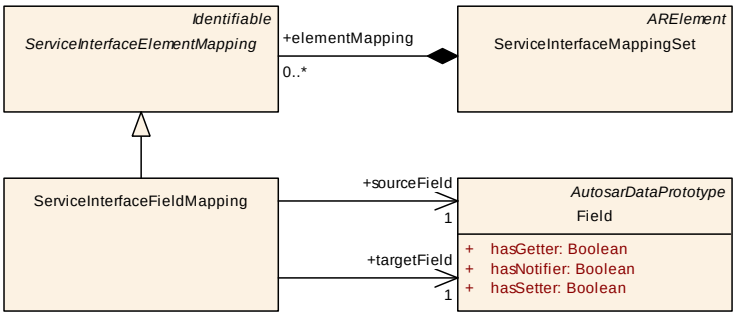


Figure 3.29: Modeling of the `ServiceInterfaceFieldMapping`

Class	**ServiceInterfaceFieldMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping			
Note	This meta-class allows to define a mapping between fields of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceInterfaceElementMapping*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sourceField	Field	1	ref	Reference to a field that is contained in the source ServiceInterface. **Tags:**atp.Status=draft
targetField	Field	1	ref	Reference to a field that is contained in the composite ServiceInterface. **Tags:**atp.Status=draft

Table 3.38: ServiceInterfaceFieldMapping

The explicit mapping implemented by `ServiceInterfaceFieldMapping` does **not** require equal `shortName`s on both sides of the mapping.

It is also possible to map a given `field` of a given `ServiceInterface` multiple times in different roles to the `ServiceInterface` that aggregates the `targetField`, as exemplified by Figure 3.30.

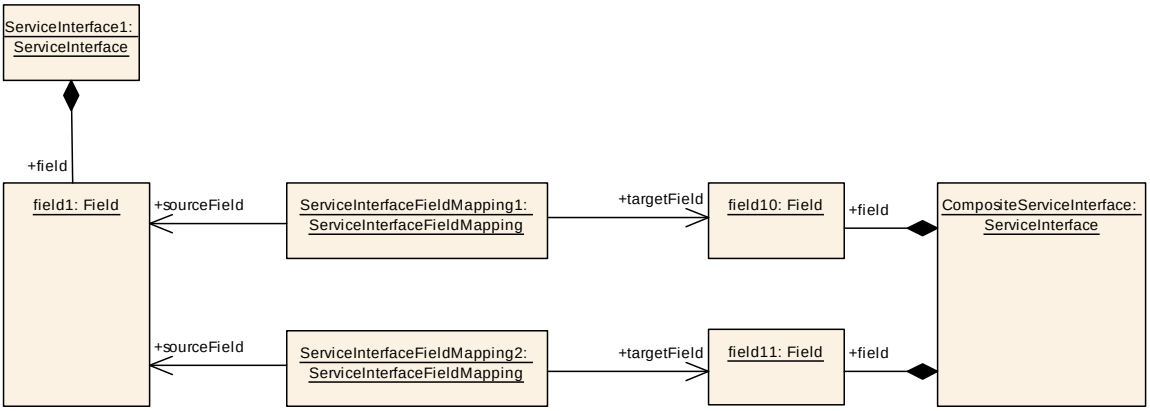


Figure 3.30: Example for the application of a `ServiceInterfaceFieldMapping`

Please note that the mapping of one `sourceField` to different `targetField`s does **not** represent a *fan-out* of any kind.

It only means that the sourceField will be used in different roles, as specified in the deployment. For more explanation, please find an example of how the role-based mapping of elements of ServiceInterfaces works in Figure A.5.

3.6.4 Service Interface Method Mapping

[TPS_MANI_01026]{DRAFT} **Semantics of ServiceInterfaceMethodMapping** ⌈Meta-class ServiceInterfaceMethodMapping has the ability to map a ServiceInterface.method referenced in the role sourceMethod explicitly to another ServiceInterface.method referenced in the role targetMethod.⌋*(RS_MANI_00017)*

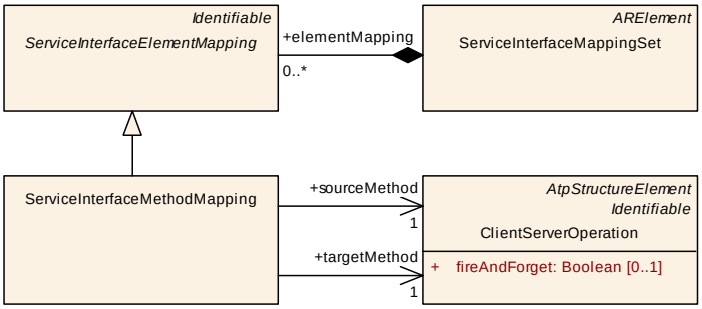


Figure 3.31: Modeling of the ServiceInterfaceMethodMapping

Class	**ServiceInterfaceMethodMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping			
Note	This meta-class allows to define a mapping between methods of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceInterfaceElementMapping*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sourceMethod	ClientServerOperation	1	ref	Reference to a method that is contained in the source ServiceInterface. **Tags:**atp.Status=draft
targetMethod	ClientServerOperation	1	ref	Reference to a method that is contained in the composite ServiceInterface. **Tags:**atp.Status=draft

Table 3.39: ServiceInterfaceMethodMapping

The explicit mapping implemented by ServiceInterfaceMethodMapping does **not** require equal shortNames on both sides of the mapping.

It is also possible to map a given method of a given ServiceInterface multiple times in different roles to the ServiceInterface that aggregates the targetMethod, as exemplified by Figure 3.32.

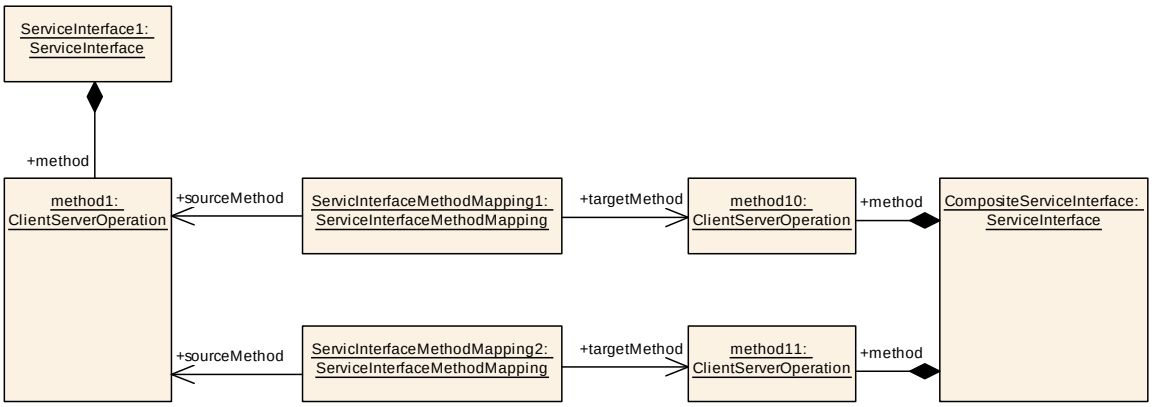


Figure 3.32: Example for the application of a `ServiceInterfaceMethodMapping`

Please note that the mapping of one `sourceMethod` to different `targetMethod`s does **not** represent a *fan-out* of any kind.

It only means that the `sourceMethod` will be used in different roles, as specified in the deployment. For more explanation, please find an example of how the role-based mapping of elements of `ServiceInterface`s works in Figure A.5.

3.7 Service Needs

3.7.1 Overview

The vast majority of use cases for `ServiceNeeds` is applicable to the *AUTOSAR classic platform* and documented in the TPS Software Component Template [1].

However, as explained in section 4.1, there are also some case where `ServiceNeeds` can be successfully used also on the *AUTOSAR adaptive platform*.

For this purpose it is possible to reuse `ServiceNeeds` defined on the *AUTOSAR classic platform*. However, there are some use cases for the application of very specific subclasses of `ServiceNeeds` that are not available on the *AUTOSAR classic platform*.

The missing subclasses of meta-class `ServiceNeeds` are defined in this chapter.

3.7.2 Service Needs for Diagnostics

The introduction of the `DiagnosticPortInterface` (see section 3.11) and extensions to the `DiagnosticMapping` (as explained in section 4.1) for the purpose of implementing diagnostic communication on the *AUTOSAR adaptive platform* it is necessary to introduce further subclasses of ServiceNeeds.

Please note that this chapter contains a description of use cases for the diagnostic `ServiceNeeds`. The description looks very similar to the corresponding descriptions in the TPS Software Component Template.

The difference, however, is that the value of RoleBasedPortAssignment.role in the TPS Software Component Template describes the name of a PortInterface modeled on the M1 level while this chapter uses the names of meta-classes on M2.

[TPS_MANI_01256]{DRAFT} **AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticIndicatorInterface** ⌈The aggregation of a DiagnosticIndicatorNeeds at a given SwcServiceDependency indicates a service use case where the application software implements a warning indicator.

ServiceNeeds kind DiagnosticIndicatorNeeds

RoleBasedPortAssignment valid roles:

- DiagnosticIndicatorInterface [1]

RoleBasedDataAssignment
N/A

RepresentedPortGroups
N/A

⌋*(RS_MANI_00061)*

Class	**DiagnosticIndicatorNeeds**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::DiagnosticServiceNeeds			
Note	This meta-class represents the needs of a software-component to provide the capability to implement an indicator. **Tags:**atp.Status=draft			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.40: DiagnosticIndicatorNeeds

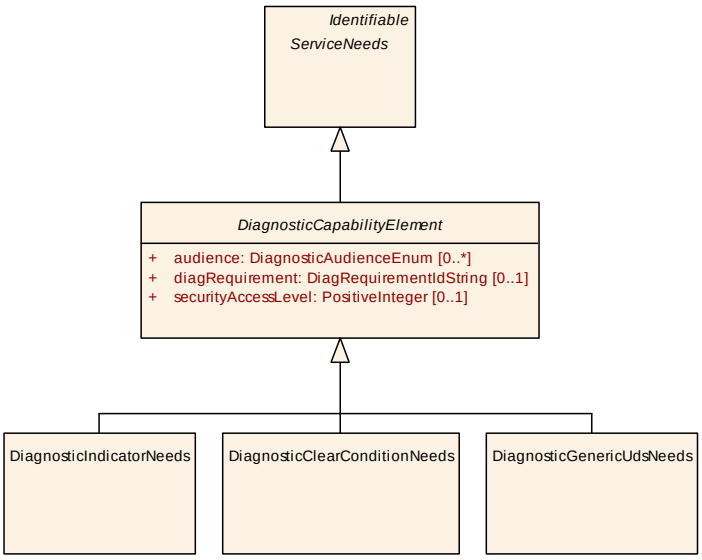


Figure 3.33: Modeling of diagnostic ServiceNeeds specifically for the *AUTOSAR adaptive platform*

[TPS_MANI_01257]{DRAFT} **AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticConditionInterface** ⌈The aggregation of a DiagnosticClearConditionNeeds at a given SwcServiceDependency indicates a service use case where the application software implements a clear condition that can be queried by the Diagnostic Manager.

ServiceNeeds kind DiagnosticClearConditionNeeds

RoleBasedPortAssignment valid roles:

- DiagnosticConditionInterface [1]

RoleBasedDataAssignment
N/A

RepresentedPortGroups
N/A

⌋*(RS_MANI_00061)*

Class	**DiagnosticClearConditionNeeds**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::DiagnosticServiceNeeds			
Note	This meta-class represents the needs of a software-component to provide the capability to set a clear condition. **Tags:**atp.Status=draft			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.41: DiagnosticClearConditionNeeds

[TPS_MANI_01258]{DRAFT} **AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticGenericUdsInterface** ⌈The aggregation of a DiagnosticGenericUdsNeeds at a given SwcServiceDependency indicates a service use case where the application software implements a generic handler of UDS services.

ServiceNeeds kind DiagnosticGenericUdsNeeds

RoleBasedPortAssignment valid roles:

- DiagnosticGenericUdsInterface [1]

RoleBasedDataAssignment
N/A

RepresentedPortGroups
N/A

⌋*(RS_MANI_00061)*

Class	**DiagnosticGenericUdsNeeds**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::DiagnosticServiceNeeds			
Note	This meta-class represents the needs of a software-component to provide the capability to process a generic UDS service. **Tags:**atp.Status=draft			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.42: DiagnosticGenericUdsNeeds

3.8 Persistency Interface

3.8.1 Overview

3.8.1.1 The big Picture

The *AUTOSAR adaptive platform* foresees a support for access to persistent data by e.g. application software.

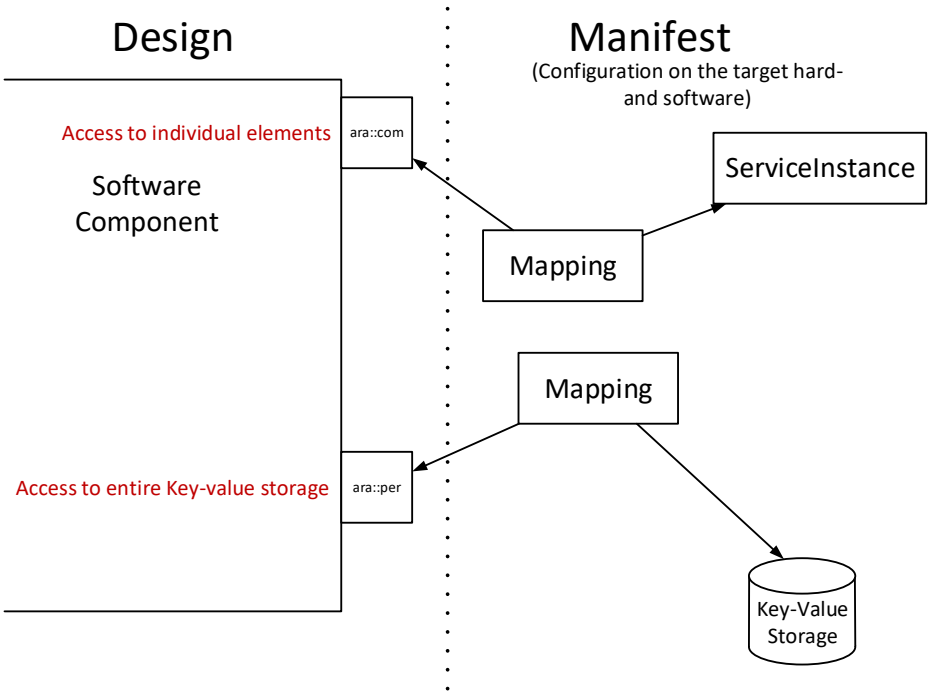


Figure 3.34: General approach for the modeling of persistency

There are some similarities to the communication model in terms of the usage of `PortPrototype`s.

In contrast to the configuration of communication, however, the modeling approach is much less detailed (i.e. instead of providing access to individual elements of a key-value storage an entire key-value storage is accessible on the level of PortPrototype).

The aspect of deployment for the configuration of persistent data is explained in Figure 3.34.

Please note that the AUTOSAR meta-model actually defines two separate meta-classes (for more details, please refer to Figure 3.35) for the different use cases of access to persistent data (i.e. PersistencyKeyValueStorageInterface) and access to files on the file system, or maybe an emulation of one (by means of PersistencyFileStorageInterface).

3.8.1.2 Modeling of Persistency Interface

Abstract meta-class PersistencyInterface has been created as a means of categorization, i.e. it allows for easily referring to PortInterfaces dedicated to persistency in general.

Class	***PersistencyInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the abstract ability to define a PortInterface for the support of persistency use cases. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	PersistencyFileStorageInterface, PersistencyKeyValueStorageInterface			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
minimum SustainedSize	PositiveInteger	0..1	attr	The value of this attribute represents the minimum size required at design time for the enclosing Persistency Interface.
redundancy	PersistencyRedundancy Enum	0..1	attr	This attribute represents a requirement towards the redundancy of storage.
redundancy Handling	PersistencyRedundancy Handling	*	aggr	This aggregation represents the chosen approaches to handle redundancy for the various use cases implemented by subclasses **Tags:**atp.Status=draft
updateStrategy	PersistencyCollection LevelUpdateStrategy Enum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyInterface as a whole.

Table 3.43: PersistencyInterface

As a counterpart to the abstract base class PersistencyInterface on interface level, meta-class PersistencyInterfaceElement has been defined as an abstract base class for elements of a PersistencyInterface.

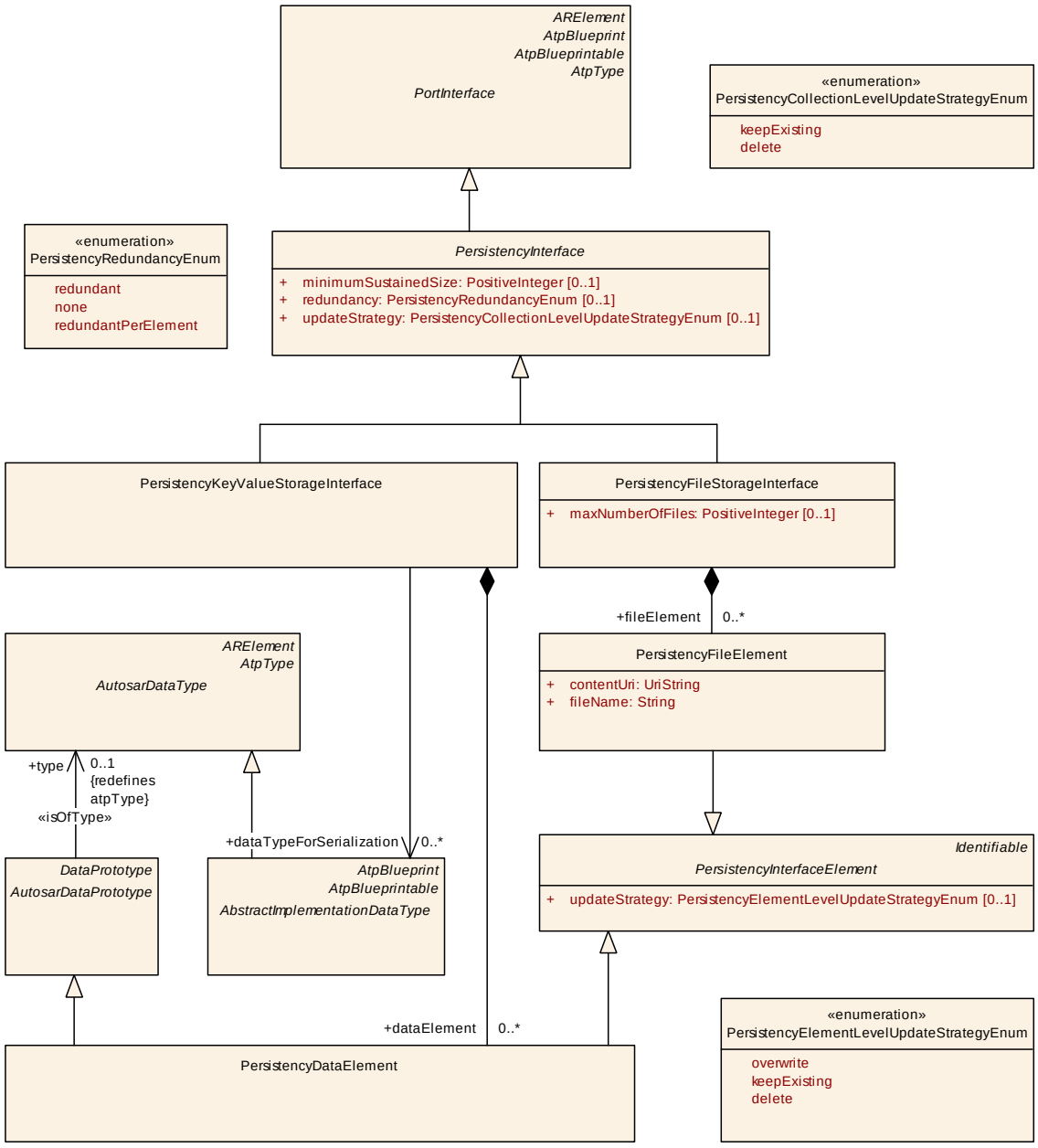


Figure 3.35: Specification of `PortInterface`s for persistency use cases

Class	***PersistencyInterfaceElement*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the abstract ability to define an element of a PortInterface for the support of persistency use cases. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	PersistencyDataElement, PersistencyFileElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	***PersistencyInterfaceElement*** (abstract)			
updateStrategy	PersistencyElement LevelUpdateStrategy Enum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyInterfaceElement.

Table 3.44: PersistencyInterfaceElement

[TPS_MANI_01194]{DRAFT} **Semantics of `PersistencyInterface.minimumSustainedSize`** ⌈Attribute `PersistencyInterface.minimumSustainedSize` can be used for the definition of a minimum amount of storage that the `PersistencyInterface` will need to allocate from the application designer's point of view.⌋ *(RS_MANI_00027)*

3.8.1.3 Redundancy Handling

[TPS_MANI_01204]{DRAFT} **Specification of redundancy of persistent data** ⌈The attribute `PersistencyInterface.redundancy` can be taken to specify whether the respective key-value storage or file storage shall store data redundantly from the perspective of the designer of the software-component.⌋ *(RS_MANI_00027)*

The details are left to an integrator who may also decide to overrule the value of `PersistencyInterface.redundancy` entirely if there is a use case for that.

Enumeration	**PersistencyRedundancyEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec
Note	This meta-class provides a way to specify in which way redundancy shall be applied on collection level. **Tags:**atp.Status=draft
Literal	***Description***
none	This value represents the requirement that redundancy measures are not applied on persistency storage level. **Tags:**atp.EnumerationLiteralIndex=1
redundant	This value represents the requirement that redundancy measures are applied on persistency storage level. The nature of the redundant persistent storage is not further qualified and subject to integrator decisions. **Tags:**atp.EnumerationLiteralIndex=0
redundantPer Element	This value represents the requirement that redundancy measures are applied on key-value level of a key-value storage or on file level of a file storage. The nature of the redundancy used on the persistent storage is not further qualified and subject to integrator decisions. **Tags:**atp.EnumerationLiteralIndex=2

Table 3.45: PersistencyRedundancyEnum

[TPS_MANI_01319]{DRAFT} **Modeling of redundancy in the context of `PersistencyInterface`** ⌈As an alternative to the ability to use `PersistencyInterface.redundancy` for announcing the consideration of redundancy at all, the design level

for persistency also provides the ability to provide a more detailed definition of redundant behavior for both key-value storage and files by means of the aggregation of PersistencyRedundancyHandling at PersistencyInterface.

This modeling is attached to the abstract base class PersistencyInterface in order to let both aspects of persistency (i.e. key-value storage and file storage) on the AUTOSAR adaptive platform benefit from the existence of meta-class PersistencyRedundancyHandling.⌋(*RS_MANI_00027*)

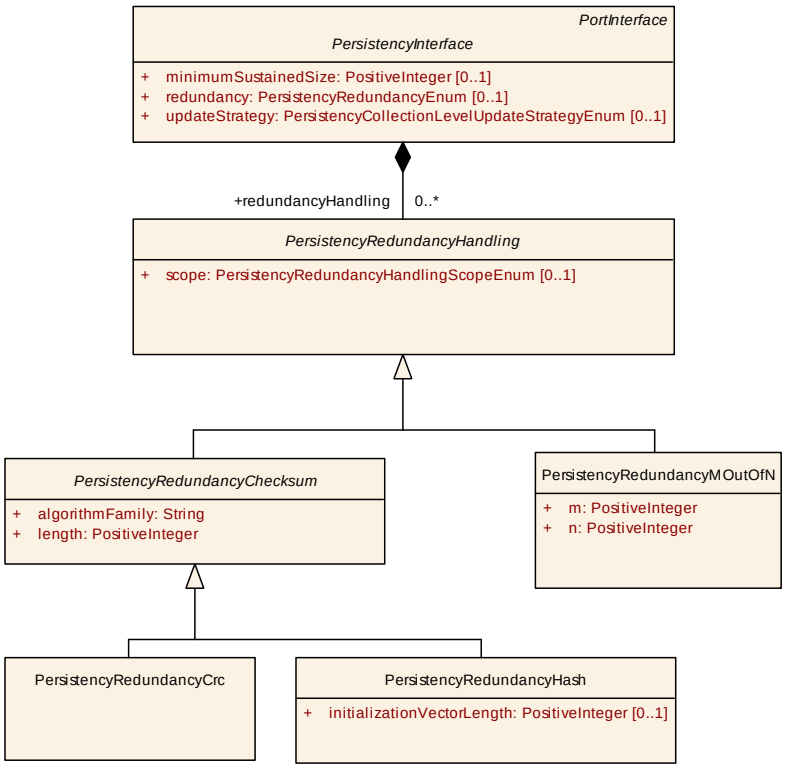


Figure 3.36: Specification of redundancy on the level of PersistencyInterface

Class	***PersistencyRedundancyHandling*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This abstract base class represents a formal description of redundancy. **Tags:**atp.Status=draft			
Base	*ARObject*			
Subclasses	*PersistencyRedundancyChecksum*, PersistencyRedundancyMOutOfN			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
scope	PersistencyRedundancyHandlingScopeEnum	0..1	attr	This attribute controls the scope in which the redundancy handling is applied.

Table 3.46: PersistencyRedundancyHandling

[TPS_MANI_01320]{DRAFT} **Definition of redundancy on interface level may be overruled in deployment** ⌈The modeling of redundancy by means of PersistencyInterface.redundancyHandling represents the intention of the designer of the PersistencyInterface.

While this is certainly a valuable input to the deployment phase, it is explicitly foreseen that an integrator may overrule the design decision regarding persistency based on superior knowledge only available at deployment time.⌋*(RS_MANI_00027)*

[constr_1746]{DRAFT} **Mutual exclusive existence of `PersistencyInterface.redundancy` and `PersistencyInterface.redundancyHandling`** ⌈For each `PersistencyInterface`, either the attribute `redundancy` or the aggregation of `PersistencyRedundancyHandling` in the role `redundancyHandling` may exist.⌋ *()*

Enumeration	**PersistencyRedundancyHandlingScopeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency
Note	This meta-class provides values to control the scope of redundancy measures in the persistency deployment **Tags:**atp.Status=draft
Literal	***Description***
persistency Redundancy HandlingScope Element	The redundancy handling shall be applied on element level (key-value pair and file). **Tags:**atp.EnumerationLiteralIndex=0
persistency Redundancy HandlingScope Storage	The redundancy handling shall be applied on storage (key-value storage and file storage) level. **Tags:**atp.EnumerationLiteralIndex=1

Table 3.47: PersistencyRedundancyHandlingScopeEnum

[TPS_MANI_01207]{DRAFT} **Standardized values of attribute `PersistencyRedundancyChecksum.algorithmFamily`** ⌈The following values of attribute `PersistencyRedundancyChecksum.algorithmFamily` are standardized by AUTOSAR:

- **`CRC_J1850`**
- **`CRC_CCITT_FALSE`**
- **`CRC_ETHERNET`**
- **`CRC_0x42F0E1EBA9EA3693`**
- **`CRC_8H2F`**
- **`CRC_16ARC`**
- **`CRC_32P4`**

⌋*(RS_MANI_00027)*

[constr_1668]{DRAFT} **Allowed combinations of `PersistencyRedundancyChecksum.length` and `algorithmFamily`** ⌈The allowed combinations of `PersistencyRedundancyChecksum.length` and `algorithmFamily` are documented in Table 3.48.⌋ *()*

	8	16	32	64
CRC_J1850	X			
CRC_CCITT_FALSE		X		
CRC_ETHERNET			X	
CRC_0x42F0E1EBA9EA3693				X
CRC_8H2F	X			
CRC_16ARC		X		
CRC_32P4			X	

Table 3.48: Allowed combinations of PersistencyRedundancyChecksum.length and algorithmFamily

Class	***PersistencyRedundancyChecksum*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	Abstract class that defines the common attributes for implementations of redundancy. **Tags:**atp.Status=draft			
Base	*ARObject*, *PersistencyRedundancyHandling*			
Subclasses	PersistencyRedundancyCrc, PersistencyRedundancyHash			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
algorithmFamily	String	1	attr	This attribute identifies the algorithm family that is used to execute the CRC/Hash.
length	PositiveInteger	1	attr	This attribute describes the length of the CRC/Hash in the unit bits.

Table 3.49: PersistencyRedundancyChecksum

Class	**PersistencyRedundancyCrc**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class formally describes the usage of a CRC for the implementation of redundancy. **Tags:**atp.Status=draft			
Base	*ARObject*, *PersistencyRedundancyChecksum*, *PersistencyRedundancyHandling*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.50: PersistencyRedundancyCrc

Class	**PersistencyRedundancyHash**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class formally describes the usage of a Hash for the implementation of redundancy. **Tags:**atp.Status=draft			
Base	*ARObject*, *PersistencyRedundancyChecksum*, *PersistencyRedundancyHandling*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initialization VectorLength	PositiveInteger	0..1	attr	Length of the initialization vector.

Table 3.51: PersistencyRedundancyHash

[constr_1751]{DRAFT} **Value of `PersistencyRedundancyMOutOfN.n` and `PersistencyRedundancyMOutOfN.m`** ⌈The value of attribute `PersistencyRedundancyMOutOfN.m` shall be set at least to 2 and at most to the value of attribute `PersistencyRedundancyMOutOfN.n`, i.e. the allowed interval is [2..`PersistencyRedundancyMOutOfN.n`].⌋*()*

Class	**PersistencyRedundancyMOutOfN**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class provides the ability to describe redundancy via an "M out of N" approach. In this case N is the number of copies created and M is the minimum number of identical copies to justify a reliable read access to the data. **Tags:**atp.Status=draft			
Base	*ARObject*, *PersistencyRedundancyHandling*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
m	PositiveInteger	1	attr	This attribute represents the "M" coordinate in the "M out of N" scheme.
n	PositiveInteger	1	attr	This attribute represents the "N" coordinate in the "M out of N" scheme.

Table 3.52: PersistencyRedundancyMOutOfN

3.8.1.4 Update Handling

[TPS_MANI_01139]{DRAFT} **Semantics of `PersistencyInterface.updateStrategy`** ⌈The attribute `PersistencyInterface.updateStrategy` can be used to specify the strategy for updating the actual persistent elements used in the context of the `PersistencyDeployment` that corresponds to `PersistencyInterface`.

This update strategy shall be applied to the `PersistencyInterface` as a whole except for the explicitly modeled `PersistencyInterfaceElement`s that define their own `updateStrategy`.⌋*(RS_MANI_00027)*

Enumeration	**PersistencyElementLevelUpdateStrategyEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface
Note	This enumeration provides possible values for the update strategy on element level. **Tags:**atp.Status=draft
Literal	***Description***
delete	The update strategy is to delete the value of the respective data item. **Tags:**atp.EnumerationLiteralIndex=2
keepExisting	The update strategy is to keep the existing value of the respective data item. **Tags:**atp.EnumerationLiteralIndex=1
overwrite	The update strategy is to overwrite the respective data item. **Tags:**atp.EnumerationLiteralIndex=0

Table 3.53: PersistencyElementLevelUpdateStrategyEnum

[TPS_MANI_01140]{DRAFT} **Semantics of `PersistencyInterfaceElement.updateStrategy`** ⌈The attribute `PersistencyInterfaceElement.updateStrategy` can be used to specify the strategy for updating the actual persistent element that corresponds to `PersistencyInterfaceElement`.⌋*(RS_MANI_00027)*

Enumeration	**PersistencyCollectionLevelUpdateStrategyEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface
Note	This enumeration provides possible values for the update strategy on interface/storage level. **Tags:**atp.Status=draft
Literal	***Description***
delete	The update strategy is to delete all values on the level of the respective collection. **Tags:**atp.EnumerationLiteralIndex=1
keepExisting	The update strategy is to keep the existing values on the level of the respective collection. **Tags:**atp.EnumerationLiteralIndex=0

Table 3.54: PersistencyCollectionLevelUpdateStrategyEnum

The behavior of the software in terms of applying an update strategy is explained in detail in [9].

3.8.2 Persistency Key Value Storage Interface

[TPS_MANI_01065]{DRAFT} **Purpose of `PersistencyKeyValueStorageInterface`** ⌈The purpose of the `PersistencyKeyValueStorageInterface` is to support the persistent access to data in a key-value storage.⌋*(RS_MANI_00027)*

Class	**PersistencyKeyValueStorageInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to implement a PortInterface for supporting persistency use cases for data. **Tags:** atp.Status=draft atp.recommendedPackage=PersistencyKeyValueStorageInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyInterface*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	PersistencyDataElement	*	aggr	This aggregation represents the collection of PersistencyDataElements in the context of the enclosing PersistencyKeyValueStorageInterface. **Tags:**atp.Status=draft
dataTypeForSerialization	AbstractImplementationDataType	*	ref	This reference identifies the AbstractImplementationDataTypes that shall be supported for storing in a key-value storage in addition to the types already determined from tha aggregation of PersistencyDataElement. **Tags:**atp.Status=draft

Table 3.55: PersistencyKeyValueStorageInterface

[TPS_MANI_01135]{DRAFT} **Semantics of PersistencyKeyValueStorageInterface.dataTypeForSerialization** ⌈The reference PersistencyKeyValueStorageInterface.dataTypeForSerialization can be taken to get information about data types for which a serialization algorithm has to be generated in order to support the persistent storage of objects of such data type.⌋*(RS_MANI_00027)*

In contrast to other kinds of PortInterfaces it is **not required** to define elements of a PersistencyKeyValueStorageInterface. If this is intended, however, the aggregation PersistencyKeyValueStorageInterface.dataElement shall be used for this purpose.

[TPS_MANI_01138]{DRAFT} **Semantics of PersistencyKeyValueStorageInterface.dataElement** ⌈By aggregating PersistencyDataElement in the role dataElement, it is possible to explicitly model key-value pairs (and some of their properties) accessible to the application software within the context of a PersistencyKeyValueStorageInterface.⌋*(RS_MANI_00027)*

[TPS_MANI_01180]{DRAFT} **Collection of data types that requires serialization support** ⌈The collection of data types that requires serialization support consists of

- AbstractImplementationDataTypes referenced in the role PersistencyKeyValueStorageInterface.dataTypeForSerialization
- either
  - AbstractImplementationDataTypes taken to type a PersistencyKeyValueStorageInterface.dataElement or
  - AbstractImplementationDataTypes mapped to ApplicationDataTypes taken to type a PersistencyKeyValueStorageInterface.dataElement by means of PortInterfaceToDataTypeMapping.dataTypeMappingSet that also refers to the enclosing PersistencyKeyValueStorageInterface.

⌋*(RS_MANI_00027)*

Class	**PersistencyDataElement**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the ability to formally specify a piece of data that is subject to persistency in the context of the enclosing PersistencyKeyValueStorageInterface. PersistencyDataElement represents also a key-value pair of the deployed PersistencyKeyValueStorage and provides an initial value. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *PersistencyInterfaceElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.56: PersistencyDataElement

Please note that a PersistencyDataElement can be typed by either an ApplicationDataType or else a CppImplementationDataType.

3.8.3 Persistency File Storage Interface

[TPS_MANI_01067]{DRAFT} **Purpose of `PersistencyFileStorageInterface`** ⌈The purpose of meta-class `PersistencyFileStorageInterface` is to support access to an abstract representation of file storage.⌋*(RS_MANI_00027)*

As far as AUTOSAR persistency is concerned, a file can have binary or text content. If it has text content then the content of the file is expected to be encoded as UTF-8 encoding with UNIX line endings.

[TPS_MANI_01068]{DRAFT} **Semantics of `PersistencyFileStorageInterface.maxNumberOfFiles`** ⌈Any `PortPrototype` typed by a `PersistencyFileStorageInterface` has the ability to access a number of files.

The upper bound of the number of files represented by a given `PortPrototype` typed by a `PersistencyFileStorageInterface` can be configured using the attribute `PersistencyFileStorageInterface.maxNumberOfFiles`.

The value of attribute `PersistencyFileStorageInterface.maxNumberOfFiles` **includes** the explicitly modeled `PersistencyFileStorageInterface.fileElements`.⌋*(RS_MANI_00027)*

Please note that the existence of the `PersistencyFileStorageInterface` does not violate the restrictions set by the POSIX subset PSE51 defined in IEEE1003.13 [10].

Class	**PersistencyFileStorageInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to implement a PortInterface for supporting persistency use cases for files. **Tags:** atp.Status=draft atp.recommendedPackage=PersistencyFileStorageInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyInterface*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
fileElement	PersistencyFileElement	*	aggr	This aggregation represents the collection of Persistency FileStorages in the context of the enclosing Persistency FileStorageInterface. **Tags:**atp.Status=draft
maxNumberOf Files	PositiveInteger	0..1	attr	This attribute represents the definition of an upper bound for the handling of files at run-time in the context of the enclosing PersistencyFileStorageInterface.

Table 3.57: PersistencyFileStorageInterface

A `PortPrototype` typed by a `PersistencyFileStorageInterface` allows for abstracting the actual calls to the operating system away from the scope of the application software and into the modules of the *AUTOSAR adaptive platform*.

[TPS_MANI_01142]{DRAFT} **Semantics of `PersistencyFileElement`** ⌈By aggregating `PersistencyFileElement` in the role `fileElement`, it is possible to explicitly model files (and some of their properties) accessible to the application software within the context of a `PersistencyFileStorageInterface`.⌋*(RS_MANI_00027)*

Class	**PersistencyFileElement**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class has the ability to represent a file at design time such that it is possible to configure the behavior for accessing the represented file at run-time. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PersistencyInterfaceElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contentUri	UriString	1	attr	This attribute represents the URI that identifies the initial content of the PersistencyFile.
fileName	String	1	attr	This attribute holds filename part of the storage location for the PersistencyFileProxy, e.g. file on the file system.

Table 3.58: PersistencyFileElement

[constr_1581]{DRAFT} **Value of `fileElement.fileName`** ⌈Within the scope of any given `PersistencyFileStorageInterface`, the value of all `fileElement.fileName` shall be unique.⌋*()*

3.9 Time Synchronization Interface

The Time Synchronization functional cluster within the Adaptive Platform is responsible to provide various Time-Base Resources for the application to read from or to write to.

In order to interface with the Time Synchronization foundation software an application developer needs to declare which kind of Time-Base Resource this application will interact with.

The interface towards the Time Synchronization follows the generic pattern of `PortPrototype`s and `PortInterface`s which are applied to many use-cases concerning the interaction of application software with platform software.

In contrast to the service based communication, the modeling of platform software interaction using `PortPrototype`s and `PortInterface`s is less detailed. The `PortPrototype` is a placeholder for the interaction with platform software, it does not model the actually used APIs available for the interaction. The APIs to be used are formally specified in the platform software SWS document, i.e. SWS_TimeSync [11].

[TPS_MANI_03535]{DRAFT} **Definition of Time Synchronization interaction** ⌈The meta-class `AbstractSynchronizedTimeBaseInterface` together with its sub classes are used to define the interaction of the application software with a Time Synchronization Time Base.⌋*(RS_MANI_00040)*

For more information, pleaser refer to Figure 3.37.

By defining an RPortPrototype which is typed by one of the AbstractSynchronizedTimeBaseInterface sub classes the application indicates that it will access a specific Time Base.

[TPS_MANI_03549]{DRAFT} **Usage of PortPrototype for the interaction with Time Synchronization** ⌈Depending on the use-case the usage of RPortPrototype or PPortPrototype typed by one of the sub-classes of AbstractSynchronizedTimeBaseInterface shall be used for the interaction with the Time Synchronization.⌋*(RS_MANI_00040)*

The application software may take the active or the passive role in the interaction with functional cluster, thus either a RPortPrototype or a PPortPrototype shall be used to represent this interaction from the application software point of view. The TimeBase Resource instance is identified using the *InstanceSpecifier* of the respective PortPrototype.

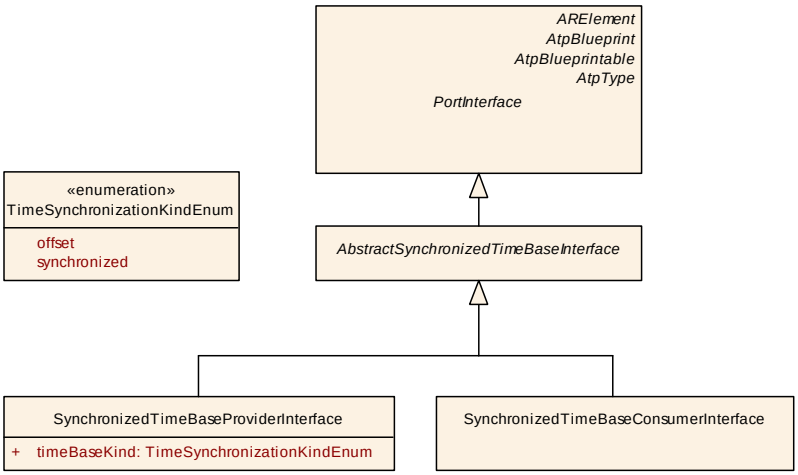


Figure 3.37: Modeling of Time Synch Interfaces

[TPS_MANI_03536]{DRAFT} **Time Synchronization interaction in a provider role** ⌈The meta-class SynchronizedTimeBaseProviderInterface is used to indicate the intended interaction with a synchronized global Time Base in a *provider* role.⌋*(RS_MANI_00040)*

When interacting with a synchronized global Time Base in a *provider* role, the application is able to *set* (and *get*) the value of the synchronized global Time Base which is then propagated to the time value on the network.

[TPS_MANI_03537]{DRAFT} **Time Synchronization interaction in a consumer role** ⌈The meta-class SynchronizedTimeBaseConsumerInterface is used to indicate the intended interaction with a synchronized global Time Base in a *consumer* role.⌋*(RS_MANI_00040)*

When interacting with a synchronized global Time Base in a *consumer* role, the application is able to only *get* the value of the synchronized global Time Base which is synchronized from a time value coming from the network.

[TPS_MANI_03551]{DRAFT} **Definition of Time Base kind** ⌈The attributes `SynchronizedTimeBaseProviderInterface.timeBaseKind` defines whether the Time Base shall be a `synchronized` or an `offset` Time Base.⌋*(RS_MANI_00040)*

Class	**SynchronizedTimeBaseProviderInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to define a PortInterface for the interaction with a Time Synchronization Provider. **Tags:** atp.Status=draft atp.recommendedPackage=TimeSynchronizationInterfaces			
Base	*ARElement*, *ARObject*, *AbstractSynchronizedTimeBaseInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
timeBaseKind	TimeSynchronization KindEnum	1	attr	Defines which kind of time base is requested at this interface. **Tags:**atp.Status=draft

Table 3.59: SynchronizedTimeBaseProviderInterface

Class	**SynchronizedTimeBaseConsumerInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to define a PortInterface for the interaction with a Time Synchronization Consumer. **Tags:** atp.Status=draft atp.recommendedPackage=TimeSynchronizationInterfaces			
Base	*ARElement*, *ARObject*, *AbstractSynchronizedTimeBaseInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.60: SynchronizedTimeBaseConsumerInterface

Enumeration	**TimeSynchronizationKindEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface
Note	Defines the possible kinds of TimeSynchronizationInterfaces. **Tags:**atp.Status=draft
Literal	***Description***
offset	Defines that the requested time base shall be an offset time based. **Tags:**atp.EnumerationLiteralIndex=1
synchronized	Defines that the requested time base shall be a synchronized time based. **Tags:**atp.EnumerationLiteralIndex=0

Table 3.61: TimeSynchronizationKindEnum

In the example in figure 3.38 the interaction of one Application with several time sync aspects are illustrated.

The interaction approach is that, for each `PortPrototype` typed by a sub-class of `AbstractSynchronizedTimeBaseInterface`, the application developer gains access to the respective kind of Time-Base Resource.

In the application code, the respective Time Base class (as defined in [11]) is constructed using the *InstanceSpecifier* representing the `PortPrototype` name.

During application deployment, those `PortPrototype`s are mapped to actual Time-Base Resources in the Time-Sync Management (see figure 9.21).

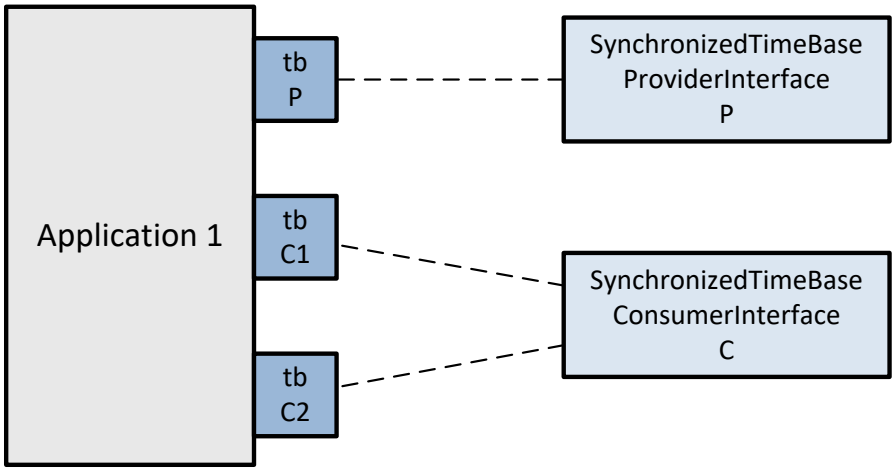


Figure 3.38: Example Application and Time Sync interaction

3.10 Platform Health Management Interface

3.10.1 Overview

Platform Health Management functional cluster within the Adaptive Platform is responsible to supervise the execution of applications, monitor their status, and triggering the State Management for respective actions.

In order to interface with the Platform Health Management foundation software an application developer needs to declare which supervisions and status information is provided by the application software and shall be observed by the Platform Health Management.

The interface towards the Platform Health Management follows the generic pattern of `PortPrototype`s and `PortInterface`s which are applied to many use-cases concerning the interaction of application software with platform software.

In contrast to the service based communication, the modeling of platform software interaction using `PortPrototype`s and `PortInterface`s is less detailed. The `PortPrototype` is a placeholder for the interaction with platform software, it does

not model the actually used APIs available for the interaction. The APIs to be used are formally specified in the platform software SWS document [12].

3.10.2 Supervised Entities and Checkpoints

The interaction of supervision with the Platform Health Management is defined by `PhmSupervisedEntityInterface` and `PhmCheckpoint`s.

[TPS_MANI_03500]{DRAFT} **Definition of Platform Health Management Supervision and Checkpoints** ⌈The meta-class `PhmSupervisedEntityInterface` together with the aggregated `PhmCheckpoint` are used to define the interaction of one Supervised Entity with the Platform Health Management supervision.⌋*(RS_MANI_00032)*

By defining an `RPortPrototype` which is typed by the `PhmSupervisedEntityInterface` the application indicates that it wants to report the `checkpoint`s of this `PhmSupervisedEntityInterface`.

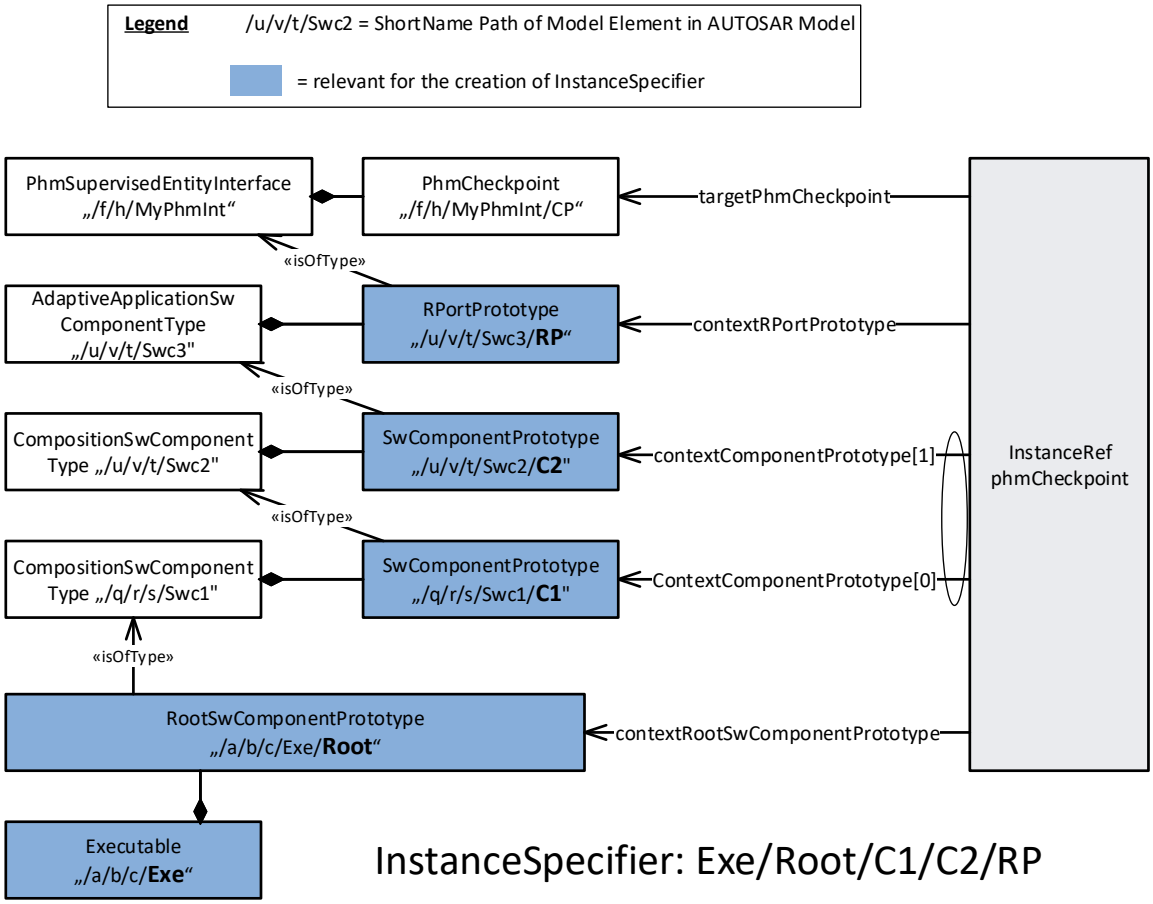


Figure 3.39: Example for the creation of an `InstanceSpecifier` of a *SupervisedEntity*

[TPS_MANI_03623]{DRAFT} **Usage of `checkpointId` in application code** ⌈The application code shall only use those `PhmCheckpoint.checkpointId` values which are defined as members of the `PhmSupervisedEntityInterface.checkpoint`.⌋ *()*

[constr_1727]{DRAFT} **Qualified combinations of `PortPrototype`s and `PhmSupervisedEntityInterface` on application software level** ⌈Within the context of an `Executable` of `category APPLICATION_LEVEL` the usage of `PhmSupervisedEntityInterface` is **only** supported for an `RPortPrototype`.⌋*()*

The application software takes the active role in the interaction with foundation platform software thus a `RPortPrototype` is used to represent this interaction from the application software point of view. The *SupervisedEntity* instance is constructed using the `InstanceSpecifier` of the respective `RPortPrototype`.

The application code then calls the *ReportCheckpoint* API (defined in [12]) of the *SupervisedEntity* (which has been constructed in the context of the respective `RPortPrototype` typed by the `PhmSupervisedEntityInterface`) in order to notify the Platform Health Management that a specific `PhmCheckpoint` has been reached in the program flow.

[constr_3530]{DRAFT} **Mandatory definition of `checkpointId`** ⌈The `checkpointId` shall be defined for every `PhmCheckpoint` element.⌋*()*

The `checkpointId` is used during the call to the *ReportCheckpoint* API as a representation of the `PhmCheckpoint`.

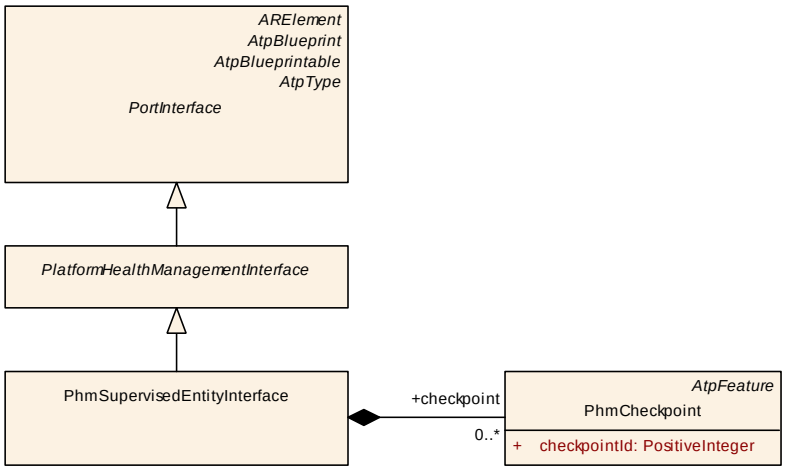


Figure 3.40: Modeling of Supervised Entities and Checkpoints

If the application wants to query the status of a Supervised Entity monitored by the Platform Health Management then the application code calls the *GetLocalSupervisionStatus* API (defined in [12]) of the *SupervisedEntity* (which has been constructed in the context of the respective `RPortPrototype` typed by the `PhmSupervisedEntityInterface`).

Note that from the application design point of view there are no relations defined between the checkpoints (as to indicate a specific observed order in reporting). The

possible transitions between the checkpoints and their timing aspects are defined in the context of the PlatformHealthManagementContribution and described in chapter 9.3.3.

Class	**PhmSupervisedEntityInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to implement a PortInterface for interaction with the Platform Health Management Supervised Entity. **Tags:** atp.Status=draft atp.recommendedPackage=PlatformHealthManagementInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PlatformHealthManagementInterface*, *Port Interface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checkpoint	PhmCheckpoint	*	aggr	Defines the set of checkpoints which can be reported on this supervised entity. **Tags:**atp.Status=draft

Table 3.62: PhmSupervisedEntityInterface

Class	**PhmCheckpoint**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to implement a checkpoint for interaction with the Platform Health Management Supervised Entity. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpFeature*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checkpointId	PositiveInteger	1	attr	Defines the numeric value which is used to indicate the reporting of this Checkpoint to the Phm. **Tags:**atp.Status=draft

Table 3.63: PhmCheckpoint

3.10.3 Health Channels

The interaction of Health Channels with the Platform Health Management is defined by PhmHealthChannelInterface and PhmHealthChannelStatus states.

[TPS_MANI_03534]{DRAFT} **Definition of Platform Health Management Health Channel** ⌈The meta-class PhmHealthChannelInterface together with the aggregated PhmHealthChannelStatus are used to define the interaction of one Health Channel with the Platform Health Management.⌋ *(RS_MANI_00032)*

By defining a RPortPrototype which is typed by the PhmHealthChannelInterface (see [constr_1728]) the application indicates that it wants to report the status of this PhmHealthChannelInterface.

The application software takes the active role in the interaction with foundation platform software thus a RPortPrototype is used to represent this interaction from the

application software point of view. The *HealthChannel* instance is constructed using the `InstanceSpecifier` of the respective `RPortPrototype`.

The application code then calls the *ReportHealthStatus* API (defined in [12]) of the *HealthChannel* (which has been constructed in the context of the respective `RPortPrototype` typed by the `PhmHealthChannelInterface`) in order to notify the Platform Health Management that the Health Channel defined by the `RPortPrototype` has changed its status.

[constr_3532]{DRAFT} **Mandatory definition of `statusId`** ⌈The `statusId` shall be defined for every `PhmHealthChannelStatus` element.⌋*()*

[TPS_MANI_03624]{DRAFT} **Usage of `statusId` in application code** ⌈The application code shall only use those `PhmHealthChannelStatus.statusId` values which are defined as members of the `PhmHealthChannelInterface.status`.⌋*()*

[TPS_MANI_03630]{DRAFT} **Semantics of `triggersRecoveryNotification`** ⌈The attribute `triggersRecoveryNotification` defines whether this specific `PhmHealthChannelStatus` shall be considered by the PHM as triggering the recovery notification.⌋*()*

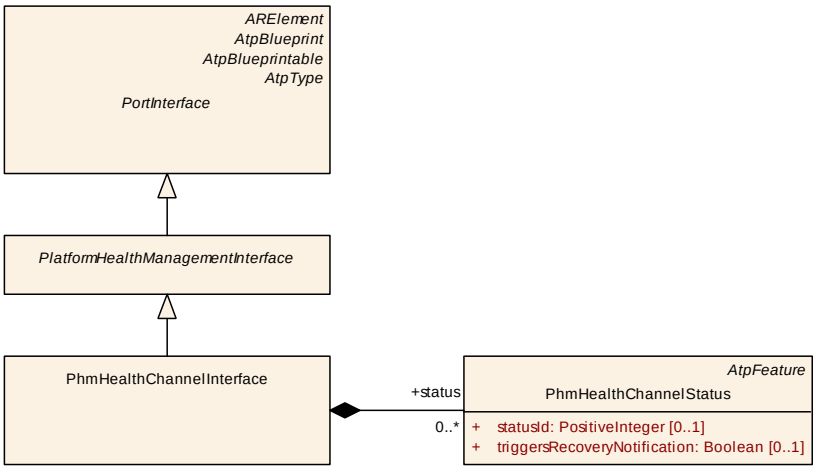


Figure 3.41: Modeling of Health Channel

[constr_1728]{DRAFT} **Qualified combinations of `PortPrototype`s and `PhmHealthChannelInterface` on application software level** ⌈Within the context of an `Executable` of `category APPLICATION_LEVEL` the usage of `PhmHealthChannelInterface` is **only** supported for a `RPortPrototype`.⌋*()*

Class	**PhmHealthChannelInterface**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface

▽

△

Class	PhmHealthChannelInterface			
Note	This meta-class provides the ability to implement a PortInterface for interaction with the Platform Health Management Health Channel. **Tags:** atp.Status=draft atp.recommendedPackage=PlatformHealthManagementInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PlatformHealthManagementInterface*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
status	PhmHealthChannelStatus	*	aggr	Defines the possible set of status information available to the health channel. **Tags:**atp.Status=draft

Table 3.64: PhmHealthChannelInterface

Class	PhmHealthChannelStatus			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	The PhmHealthChannelStatus specifies one possible status of the health channel. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpFeature*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
statusId	PositiveInteger	0..1	attr	Defines the numeric value which is used to indicate the indication of this status the Phm. **Tags:**atp.Status=draft
triggers Recovery Notification	Boolean	0..1	attr	Defines whether this PhmHealthChannelStatus shall cause the Phm to trigger the Health Channel recovery notification. True: Indicates unhealthy state. Phm to trigger the Health Channel recovery notification when the Health channel status changes to this state. False: Indicates healthy state. Phm not to trigger the Health Channel recovery notification when the Health channel status changes to this state. **Tags:**atp.Status=draft

Table 3.65: PhmHealthChannelStatus

3.10.4 Recovery notification to State Management

The Phm monitors the reporting of Supervised Entities and Checkpoints as well as the reported Health Channel status information. In case of violations the Phm can be configured to report the violation to the State Management and let the State Management deal with the recovery activities.

The example in figure 3.42 illustrates the reporting of Supervised Entities by Application 1 and 2. The Phm is configured to perform the supervision of these reported elements. In case of violations the Phm is configured to notify the State Management application to deal with the situation.

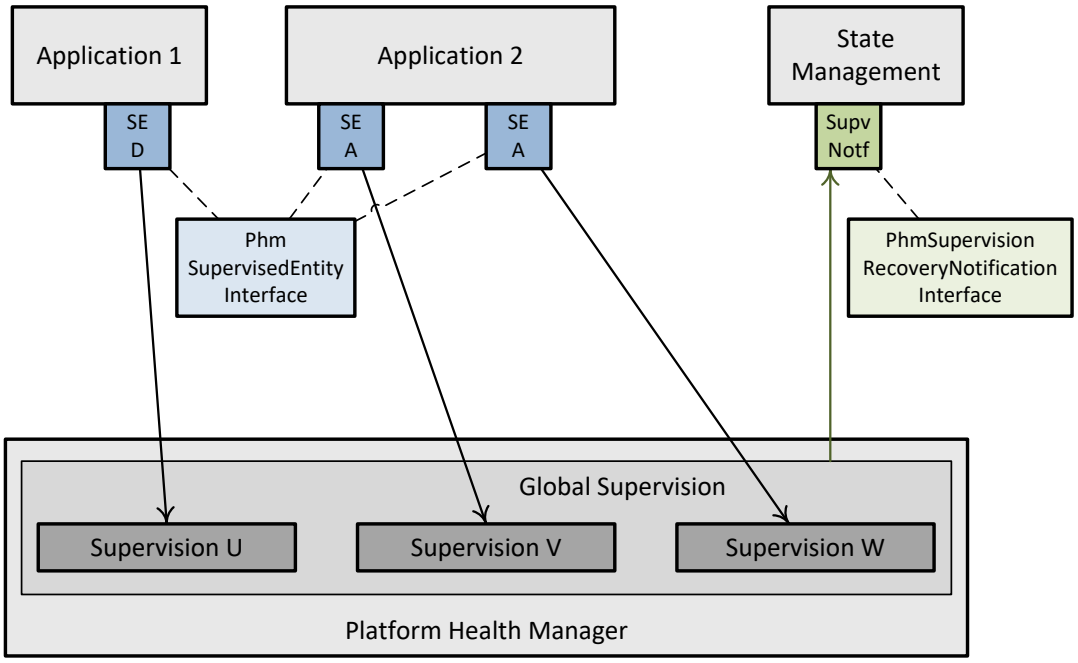


Figure 3.42: Example of a Phm monitoring and recovery setup

[TPS_MANI_01280]{DRAFT} **Semantics of meta-class `PhmSupervisionRecoveryNotificationInterface`** ⌈The recovery notification of a failed Supervision by PHM does issue is to call a piece of code on State Management software level.

The mechanism for activating the code on the level of State Management software is to model a `PPortPrototype` typed by a `PhmSupervisionRecoveryNotificationInterface`.⌋*()*

[TPS_MANI_03631]{DRAFT} **Semantics of meta-class `PhmHealthChannelRecoveryNotificationInterface`** ⌈The recovery notification of a failed HealthChannel monitoring by PHM does issue is to call a piece of code on State Management software level.

The mechanism for activating the code on the level of State Management software is to model a `PPortPrototype` typed by a `PhmHealthChannelRecoveryNotificationInterface`.⌋*()*

The operation to be called by Phm in the context of [TPS_MANI_01280] and [TPS_MANI_03631] are defined in the Platform Health Management specification document [12].

As already mentioned, the State Management is supposed to implement the recovery actions. This implies that the `PhmSupervisionRecoveryNotificationInterface` and `PhmHealthChannelRecoveryNotificationInterface` can only

be used in combination with a PPortPrototype. This aspect is clarified by [constr_1729].

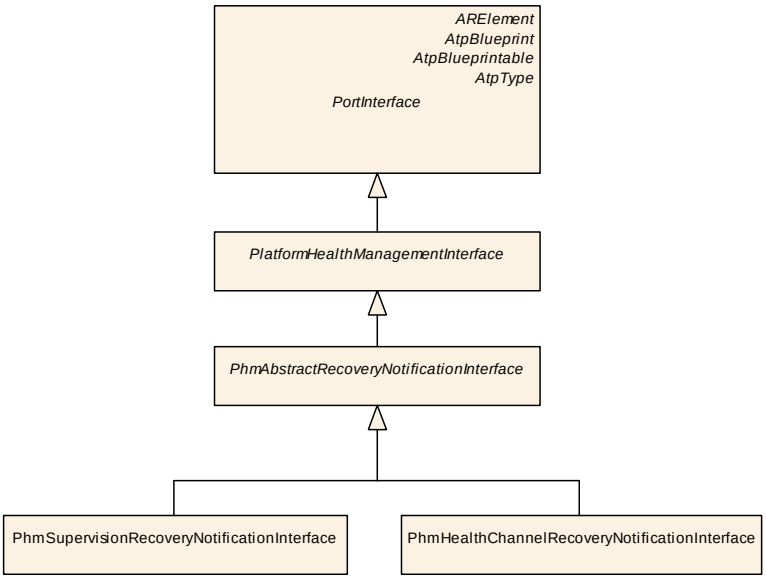


Figure 3.43: Modeling of the PhmAbstractRecoveryNotificationInterface

[constr_1729]{DRAFT} **Qualified combinations of PortPrototypes and PhmSupervisionRecoveryNotificationInterface / PhmHealthChannelRecoveryNotificationInterface on State Management software level** ⌈Within the context of an Executable of category APPLICATION_LEVEL the usage of PhmSupervisionRecoveryNotificationInterface and PhmHealthChannelRecoveryNotificationInterface is **only** supported for a PPortPrototype.⌋*()*

Class	**PhmSupervisionRecoveryNotificationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents a PortInterface that can be taken for implementing a PHM Supervision notification. **Tags:** atp.Status=draft atp.recommendedPackage=PlatformHealthManagementInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PhmAbstractRecoveryNotificationInterface*, *PlatformHealthManagementInterface*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.66: PhmSupervisionRecoveryNotificationInterface

Class	**PhmHealthChannelRecoveryNotificationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents a PortInterface that can be taken for implementing a PHM HealthChannel notification. **Tags:** atp.Status=draft atp.recommendedPackage=PlatformHealthManagementInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PhmAbstractRecoveryNotificationInterface*, *PlatformHealthManagementInterface*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.67: PhmHealthChannelRecoveryNotificationInterface

3.11 Diagnostic Interface

3.11.1 Overview

On the *AUTOSAR adaptive platform*, dedicated `PortInterface`s are defined for the interaction of application-layer software with the AUTOSAR `Diagnostic Manager`.

In contrast to the conventions on the AUTOSAR classic Platform, these `PortInterface`s and, by extension, the standardized `ara::diag` API **are only used on the application side** of this communication relation.

The interfaces on the side of the AUTOSAR `Diagnostic Manager` (and thus the part of the implementation of the `PortPrototype` that faces the AUTOSAR `Diagnostic Manager`) are **entirely proprietary**. This aspect is depicted in Figure 3.44.

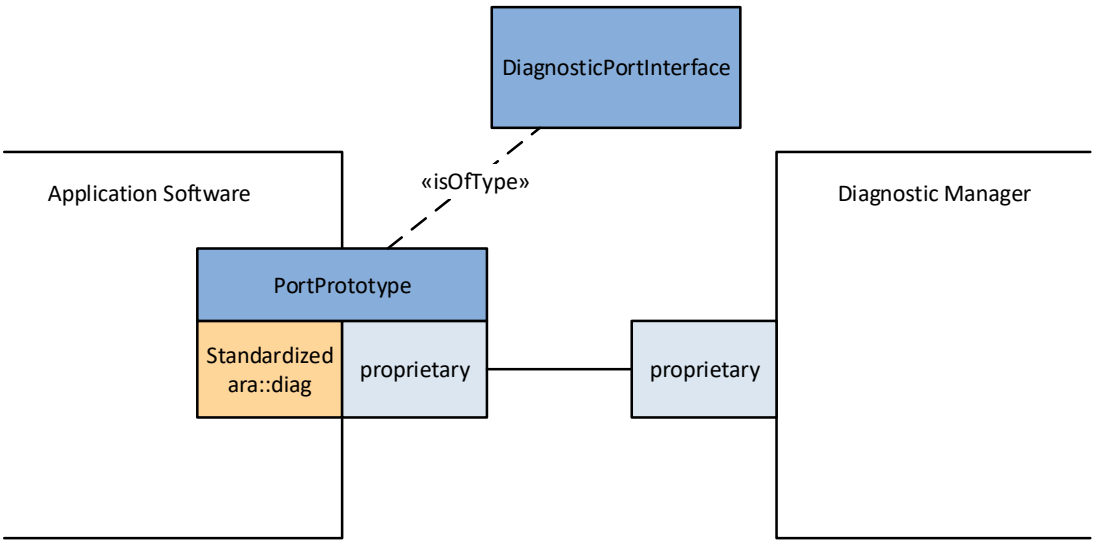


Figure 3.44: Standardized vs. proprietary parts in the implementation of `ara::diag`

This arrangement tries to provide the application programmer with the simplest possible API from the application's point of view. At the same time it hides a lot of the complexity of the interaction between application and Diagnostic Manager behind a solid abstraction layer.

[TPS_MANI_01242]{DRAFT} **PortInterfaces used for communication with the AUTOSAR Diagnostic Manager** ⌈All PortInterfaces used for this purpose are derived from the abstract meta-class DiagnosticPortInterface. A DiagnosticPortInterface does not implement a service-oriented communication pattern, in particular there is no explicit service discovery on the API level involved.⌋*(RS_MANI_00061)*

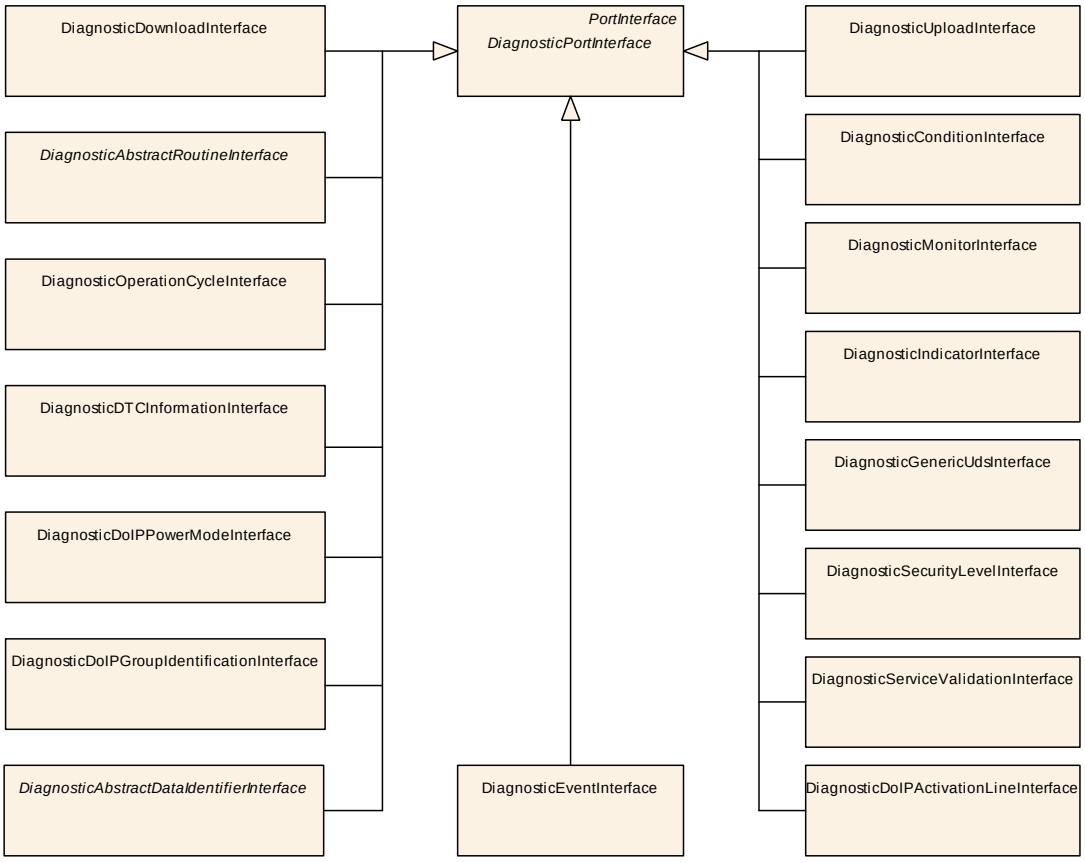


Figure 3.45: Modeling of PortInterfaces for diagnostic purposes

The specializations of DiagnosticPortInterface cover the various aspects of diagnostic communication, e.g. the implementation of diagnostic routines, the reporting of diagnostic events or the access to a Diagnostic Data Identifier (DID).

Figure 3.45 depicts all meta-classes that directly inherit from DiagnosticPortInterface.

Class	***DiagnosticPortInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class serves as an abstract base-class for all diagnostics-related PortInterfaces. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	*DiagnosticAbstractDataIdentifierInterface*, *DiagnosticAbstractRoutineInterface*, DiagnosticCondition Interface, DiagnosticDTCInformationInterface, DiagnosticDoIPActivationLineInterface, DiagnosticDoIP GroupIdentificationInterface, DiagnosticDoIPPowerModeInterface, DiagnosticDoIPTriggerVehicle AnnouncementInterface, DiagnosticDownloadInterface, DiagnosticEcuResetInterface, DiagnosticEvent Interface, DiagnosticGenericUdsInterface, DiagnosticIndicatorInterface, DiagnosticMonitorInterface, DiagnosticOperationCycleInterface, DiagnosticSecurityLevelInterface, DiagnosticServiceValidation Interface, DiagnosticUploadInterface			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.68: DiagnosticPortInterface

3.11.2 Diagnostic Routine Interface

The convention for the creation of diagnostic routines is to establish at most three methods for each diagnostic routine:

- Start the execution of the routine.
- Stop the execution of the routine.
- Request the results of the routine's execution.

In response to this convention the `DiagnosticRoutineInterface` is modeled to aggregate `ClientServerOperation` in three dedicated roles: `start`, `stop`, and `requestResult`.

[constr_1696]{DRAFT} **`ClientServerOperation` aggregated by `DiagnosticRoutineInterface`** ⌈Any `ClientServerOperation` aggregated by a `DiagnosticRoutineInterface` shall not define the following attributes:

- `fireAndForget`
- `possibleApError`
- `possibleApErrorSet`

⌋()

The arguments to the diagnostic routine shall be modeled as the arguments of the respective `ClientServerOperation`s aggregated in the roles `start`, `stop`, and `requestResult`.

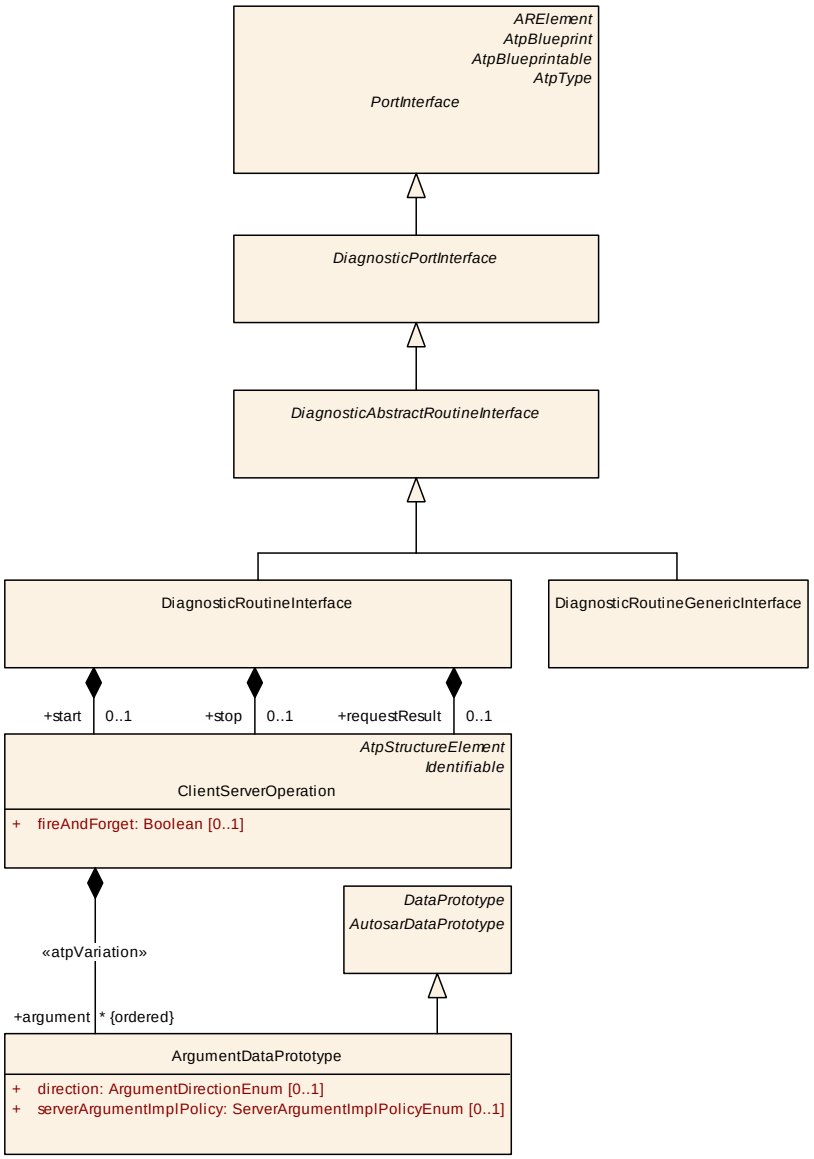


Figure 3.46: Modeling of `DiagnosticRoutineInterface`

Class	**DiagnosticRoutineInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a routine-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractRoutineInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**DiagnosticRoutineInterface**			
requestResult	ClientServerOperation	0..1	aggr	This represents the request result method of the diagnostic routine. **Tags:**atp.Status=draft
start	ClientServerOperation	0..1	aggr	This represents the start method of the diagnostic routine. **Tags:**atp.Status=draft
stop	ClientServerOperation	0..1	aggr	This represents the stop method of the diagnostic routine. **Tags:**atp.Status=draft

Table 3.69: DiagnosticRoutineInterface

In addition to the modeling of "typed" diagnostic routines using the `DiagnosticRoutineInterface` it is possible to use the `DiagnosticRoutineGenericInterface` to define a diagnostic routine for which no further formalization is provided.

Class	**DiagnosticRoutineGenericInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a generic Routine-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractRoutineInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.70: DiagnosticRoutineGenericInterface

This means that implicitly there are still up to three methods defined for the already mentioned roles of a diagnostic routine.

However, the methods inside the context of such a generic diagnostic routine would always use plain byte arrays as the arguments and therefore a formalization within the AUTOSAR meta-model does not make sense any longer.

Meta-class `DiagnosticAbstractRoutineInterface` serves as the abstract base class to all routine-related `DiagnosticPortInterface`s on the *AUTOSAR adaptive platform*.

Class	***DiagnosticAbstractRoutineInterface*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface
Note	This meta-class serves as the abstract base class of PortInterfaces dedicated to routine execution on the AUTOSAR adaptive platform. **Tags:**atp.Status=draft

▽

△

Class	**DiagnosticAbstractRoutineInterface** (abstract)			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	DiagnosticRoutineGenericInterface, DiagnosticRoutineInterface			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.71: DiagnosticAbstractRoutineInterface

3.11.3 Interface to Data Identifier and Element of Data Identifier

The ability to access diagnostic-relevant **data** in the application software is formalized in another abstract sub-class of `DiagnosticPortInterface`: `DiagnosticAbstractDataIdentifierInterface`.

Meta-class `DiagnosticAbstractDataIdentifierInterface`, in turn, defines three concrete subclasses that represent the concrete abilities to access diagnostic-related data in the application software.

[TPS_MANI_01243]{DRAFT} **Semantics of `DiagnosticDataIdentifierInterface`** ⌈`DiagnosticDataIdentifierInterface` is used to access the content of an entire DID at once.

For this purpose up to two `ClientServerOperation`s are aggregated in the roles `read` and `write`, depending on the concrete use case for a specific `DiagnosticDataIdentifierInterface`.⌋*(RS_MANI_00061)*

Class	**DiagnosticDataIdentifierInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a DID-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractDataIdentifierInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
read	ClientServerOperation	0..1	aggr	This represents the method to read the content of a diagnostic data identifier. **Tags:**atp.Status=draft
write	ClientServerOperation	0..1	aggr	This represents the method to write the contents of a diagnostic data identifier. **Tags:**atp.Status=draft

Table 3.72: DiagnosticDataIdentifierInterface

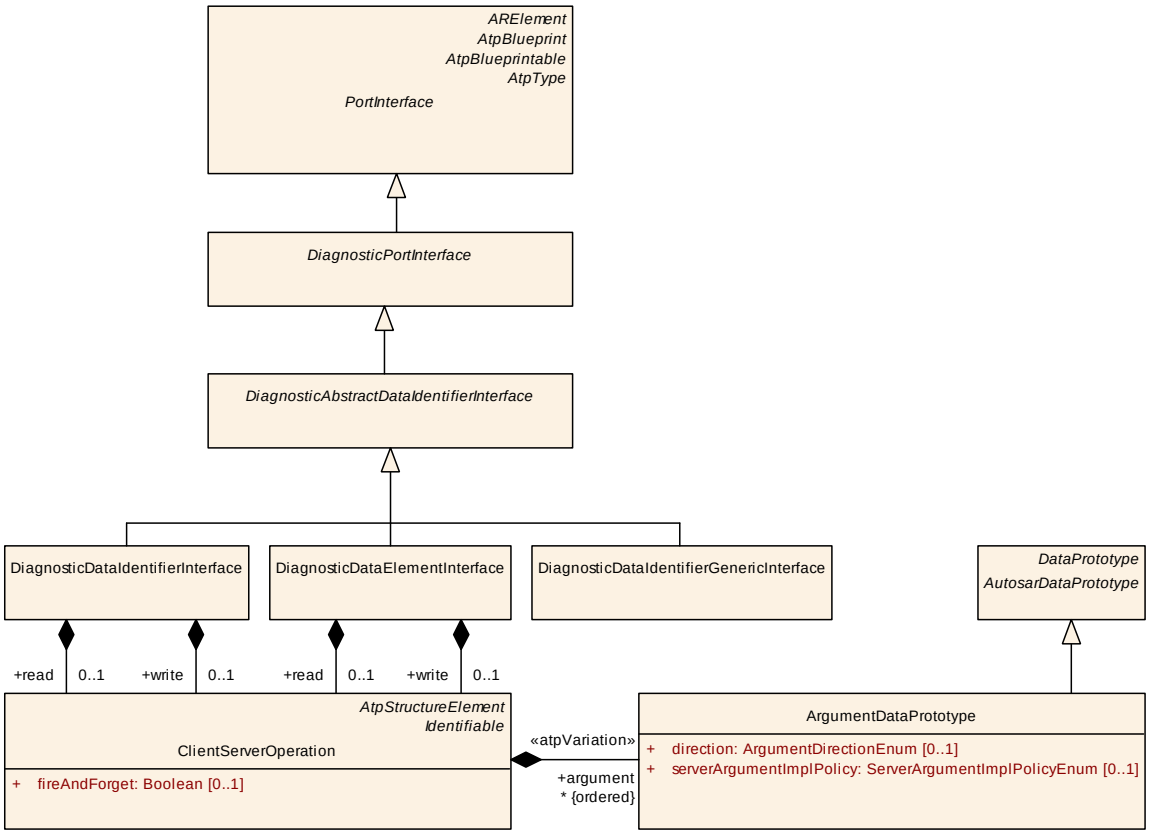


Figure 3.47: Modeling of `DiagnosticDataIdentifierInterface`

[TPS_MANI_01244]{DRAFT} **Semantics of `DiagnosticDataElementInterface`** ⌈`DiagnosticDataElementInterface` is used to access the content of an element within a given DID.

For this purpose up to two `ClientServerOperation`s are aggregated in the roles `read` and `write`, depending on the concrete use case for a specific `DiagnosticDataElementInterface`.⌋*()*

Class	**DiagnosticDataElementInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a element-of-DID-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractDataIdentifierInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
read	ClientServerOperation	0..1	aggr	This represents the method to read the content of an element of a diagnostic data identifier. **Tags:**atp.Status=draft

△

Class	DiagnosticDataElementInterface			
write	ClientServerOperation	0..1	aggr	This represents the method to write the content of an element of a diagnostic data identifier. **Tags:**atp.Status=draft

Table 3.73: DiagnosticDataElementInterface

[TPS_MANI_01245]{DRAFT} **Semantics of `DiagnosticDataIdentifierGenericInterface`** ⌈`DiagnosticDataIdentifierInterface` is used to access the content of an entire DID at once.

For this purpose methods will be defined with a read and write semantics, but these methods will always only provide arguments that are byte-arrays.

Therefore, a further formalization of these methods for reading and writing data within the context of the AUTOSAR meta-model does not make sense and is therefore omitted.⌋ *(RS_MANI_00061)*

Class	DiagnosticDataIdentifierGenericInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a generic DID-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractDataIdentifierInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.74: DiagnosticDataIdentifierGenericInterface

Please note that it is necessary to put some restrictions on the `argument` **unless** a given `DiagnosticDataIdentifierInterface` or `DiagnosticDataElementInterface` aggregates **only one** `ClientServerOperation` in **either** the role `read` **or** `write`.

[constr_1697]{DRAFT} **Restriction for `ClientServerOperation` aggregated by a `DiagnosticDataIdentifierInterface` or `DiagnosticDataElementInterface`** ⌈If meta-classes `DiagnosticDataIdentifierInterface` or `DiagnosticDataElementInterface` aggregate two `ClientServerOperation`s then

- The two `ClientServerOperation`s shall have the same number of `argument`s.
- The `argument`s on the nth position in the collection of `argument`s shall have identical properties, except the `direction`. In particular, the following conditions shall be fulfilled with respect to attribute `direction`:

– Any ArgumentDataPrototype aggregated by a ClientServerOperation that is itself aggregated in either the role DiagnosticDataIdentifierInterface.read or DiagnosticDataElementInterface.read shall set attribute direction to out.

– Any ArgumentDataPrototype aggregated by a ClientServerOperation that is itself aggregated in either the role DiagnosticDataIdentifierInterface.write or DiagnosticDataElementInterface.write shall set attribute direction to in.

⌋()

3.11.4 Interface to diagnostic Events

AUTOSAR defines several subclasses of DiagnosticPortInterface that are dedicated to the handling of diagnostic events.

[TPS_MANI_01246]{DRAFT} **Semantics of DiagnosticMonitorInterface** ⌈Meta-class DiagnosticMonitorInterface represents the ability to report diagnostic events to the AUTOSAR Diagnostic Manager.⌋*(RS_MANI_00061)*

Class	**DiagnosticMonitorInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a monitor-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.75: DiagnosticMonitorInterface

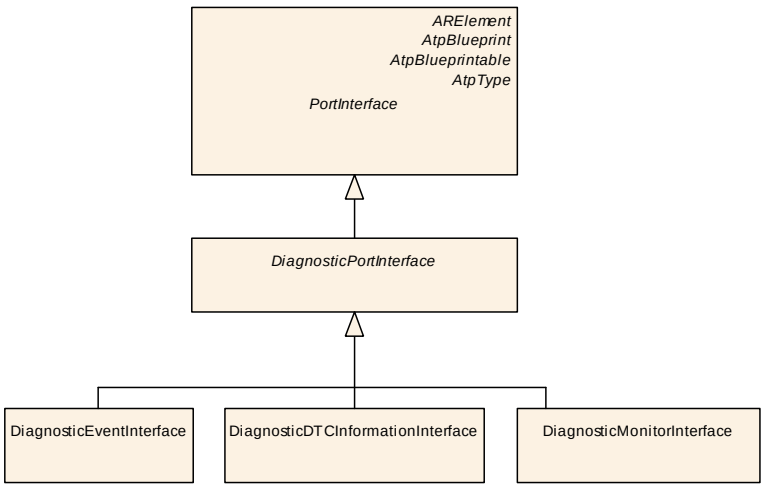


Figure 3.48: Modeling of DiagnosticEventInterface

[TPS_MANI_01247]{DRAFT} **Semantics of `DiagnosticDTCInformationInterface`** ⌈Meta-class `DiagnosticDTCInformationInterface` represents the ability to retrieve information about a given diagnostic trouble code.⌋*(RS_MANI_00061)*

Class	**DiagnosticDTCInformationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to access the properties of DTCs on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.76: DiagnosticDTCInformationInterface

[TPS_MANI_01248]{DRAFT} **Semantics of `DiagnosticEventInterface`** ⌈Meta-class `DiagnosticEventInterface` represents the ability to retrieve information about a given diagnostic event.⌋*(RS_MANI_00061)*

Class	**DiagnosticEventInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to access the properties of diagnostic events on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.77: DiagnosticEventInterface

3.11.5 Interface to diagnostic Condition

[TPS_MANI_01249]{DRAFT} **Semantics of `DiagnosticConditionInterface`** ⌈AUTOSAR supports different diagnostic conditions, i.e. enable condition and clear condition. This aspect is represented in the definition of the `DiagnosticConditionInterface` for the *AUTOSAR adaptive platform*.⌋*(RS_MANI_00061)*

The `DiagnosticConditionInterface` does not require any further details in its formalization.

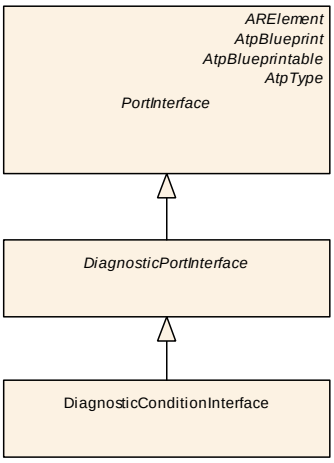


Figure 3.49: Modeling of `DiagnosticConditionInterface`

Class	**DiagnosticConditionInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for diagnostic conditions on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.78: DiagnosticConditionInterface

3.11.6 Indicator Interface

[TPS_MANI_01250]{DRAFT} **Semantics of `DiagnosticIndicatorInterface`** ⌈The usage of the `DiagnosticIndicatorInterface` is foreseen for software that implements a diagnostic indicator (i.e. a warning light on the dashboard).⌋ *(RS_MANI_00061)*

Class	**DiagnosticIndicatorInterface**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface
Note	This meta-class represents the ability to implement a PortInterface to implement indicator functionality on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces

△

Class	DiagnosticIndicatorInterface			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.79: DiagnosticIndicatorInterface

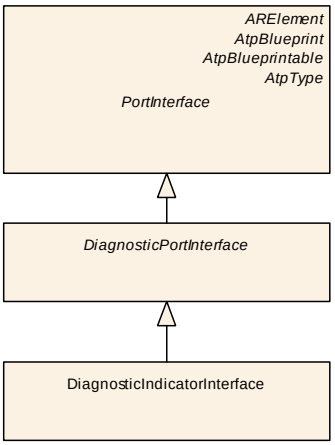


Figure 3.50: Modeling of `DiagnosticIndicatorInterface`

The `DiagnosticIndicatorInterface` does not require any further details in its formalization.

3.11.7 Security Level Interface

[TPS_MANI_01251]{DRAFT} **Semantics of `DiagnosticSecurityLevelInterface`** ⌈The usage of the `DiagnosticSecurityLevelInterface` is foreseen for software that implements the checks for the clearance of a given security level.⌋*(RS_MANI_00061)*

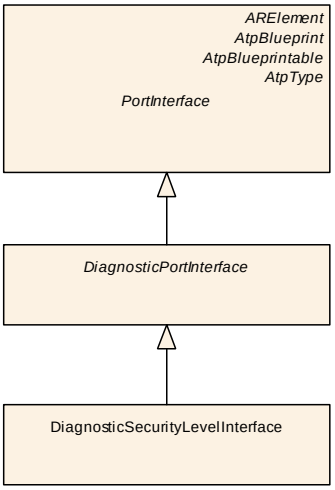


Figure 3.51: Modeling of `DiagnosticSecurityLevelInterface`

Class	**DiagnosticSecurityLevelInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a security-level-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.80: DiagnosticSecurityLevelInterface

The `DiagnosticSecurityLevelInterface` does not require any further details in its formalization.

3.11.8 Service Validation Interface

[TPS_MANI_01252]{DRAFT} **Semantics of `DiagnosticServiceValidationInterface`** ⌈The usage of the `DiagnosticServiceValidationInterface` is foreseen for software that implements the checks for clearance on manufacturer or supplier level.⌋*(RS_MANI_00061)*

The `DiagnosticServiceValidationInterface` does not require any further details in its formalization.

[TPS_MANI_01311]{DRAFT} **Handling of manufacturer checks** ⌈A manufacturer check is modeled as a `PPortPrototype` typed by a `DiagnosticServiceValidationInterface`.

The `PortPrototype` shall be referenced by a `RoleBasedPortAssignment` where attribute `role` is set to the value `DiagnosticServiceValidation`.

The `RoleBasedPortAssignment` shall be owned by a `SwcServiceDependency` that aggregates a `DiagnosticCommunicationManagerNeeds` where attribute `serviceRequestCallbackType` is set to `DiagnosticServiceRequestCallbackTypeEnum.requestCallbackTypeManufacturer`.⌋*(RS_MANI_00061)*

[TPS_MANI_01312]{DRAFT} **Handling of supplier checks** ⌈A supplier check is modeled as a `PPortPrototype` typed by a `DiagnosticServiceValidationInterface`.

The `PortPrototype` shall be referenced by a `RoleBasedPortAssignment` where attribute `role` is set to the value `DiagnosticServiceValidation`.

The `RoleBasedPortAssignment` shall be owned by a `SwcServiceDependency` that aggregates a `DiagnosticCommunicationManagerNeeds` where attribute

serviceRequestCallbackType is set to DiagnosticServiceRequestCallbackTypeEnum.requestCallbackTypeSupplier.⌋(RS_MANI_00061)

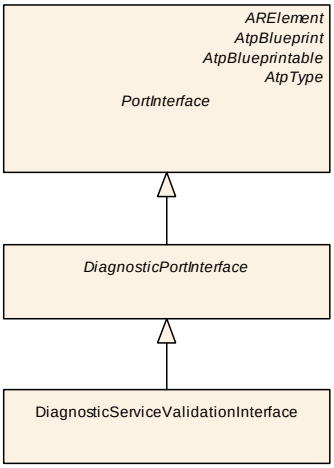


Figure 3.52: Modeling of DiagnosticServiceValidationInterface

Class	**DiagnosticServiceValidationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for service validation on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.81: DiagnosticServiceValidationInterface

The creation of a connection between the DM and any PortPrototype that fulfills [TPS_MANI_01311] and [TPS_MANI_01312] is not formalized and shall be done by the integrator.

Class	**DiagnosticCommunicationManagerNeeds**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the general needs on the configuration of the Diagnostic Communication Manager (Dcm) which are not related to a particular item (e.g. a PID or DiagnosticRoutineNeeds). The main use case is the mapping of service ports to the Dcm which are not related to a particular item.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**DiagnosticCommunicationManagerNeeds**			
serviceRequest CallbackType	DiagnosticService RequestCallbackType Enum	0..1	attr	This represents the ability to define whether the usage of PortInterface ServiceRequestNotification has the characteristics of being initiated by a manufacturer or by a supplier.

Table 3.82: DiagnosticCommunicationManagerNeeds

Enumeration	**DiagnosticServiceRequestCallbackTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
Note	This represents the ability to define whether a Service Request Notification was used in the role of a manufacturer or a supplier.
Literal	***Description***
requestCallback TypeManufacturer	This represents the case that the usage of PortInterface ServiceRequestNotification has the characteristics of being used by a manufacturer. **Tags:**atp.EnumerationLiteralIndex=0
requestCallback TypeSupplier	This represents the case that the usage of PortInterface ServiceRequestNotification has the characteristics of being used by a supplier. **Tags:**atp.EnumerationLiteralIndex=1

Table 3.83: DiagnosticServiceRequestCallbackTypeEnum

3.11.9 Operation Cycle Interface

[TPS_MANI_01253]{DRAFT} **Semantics of `DiagnosticOperationCycleInterface`** ⌈The usage of the `DiagnosticOperationCycleInterface` is foreseen for software that implements the manages the operation cycles.⌋ *(RS_MANI_00061)*

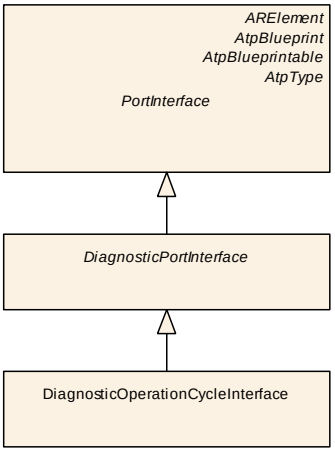


Figure 3.53: Modeling of `DiagnosticOperationCycleInterface`

Class	**DiagnosticOperationCycleInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for operation cycles on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.84: DiagnosticOperationCycleInterface

The `DiagnosticOperationCycleInterface` does not require any further details in its formalization.

3.11.10 Generic UDS Interface

[TPS_MANI_01254]{DRAFT} **Semantics of `DiagnosticGenericUdsInterface`**
⌈The AUTOSAR diagnostic communication API also foresees the existence of one DiagnosticPortInterface that support the implementation of a completely generic handler of a UDS service.⌋*(RS_MANI_00061)*

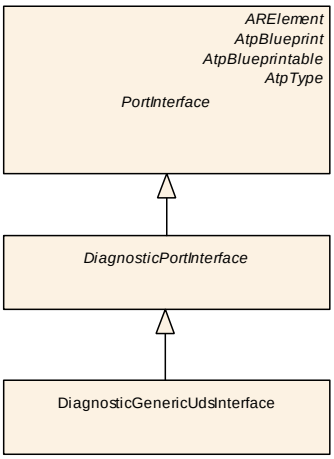


Figure 3.54: Modeling of `DiagnosticGenericUdsInterface`

Class	**DiagnosticGenericUdsInterface**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface

△

Class	**DiagnosticGenericUdsInterface**			
Note	This meta-class represents the ability to implement a generic UDS PortInterface for diagnostics on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.85: DiagnosticGenericUdsInterface

The `DiagnosticGenericUdsInterface` does not require any further details in its formalization.

3.11.11 DoIP Interfaces

[TPS_MANI_01255]{DRAFT} **Semantics of DoIP `DiagnosticPortInterface`s** ⌈The AUTOSAR diagnostic communication API also foresees the existence of `DiagnosticPortInterface`s to implement functionalities in the context of DoIP operation.

Specifically, the following concrete sub-classes of `DiagnosticPortInterface` are defined to support the implementation of functionalities in the context of DoIP:

- `DiagnosticDoIPGroupIdentificationInterface`
- `DiagnosticDoIPPowerModeInterface`
- `DiagnosticDoIPActivationLineInterface`
- `DiagnosticDoIPTriggerVehicleAnnouncementInterface`

⌋*(RS_MANI_00061)*

Class	**DiagnosticDoIPGroupIdentificationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIP Group Identification on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	DiagnosticDoIPGroupIdentificationInterface			
–	–	–	–	–

Table 3.86: DiagnosticDoIPGroupIdentificationInterface

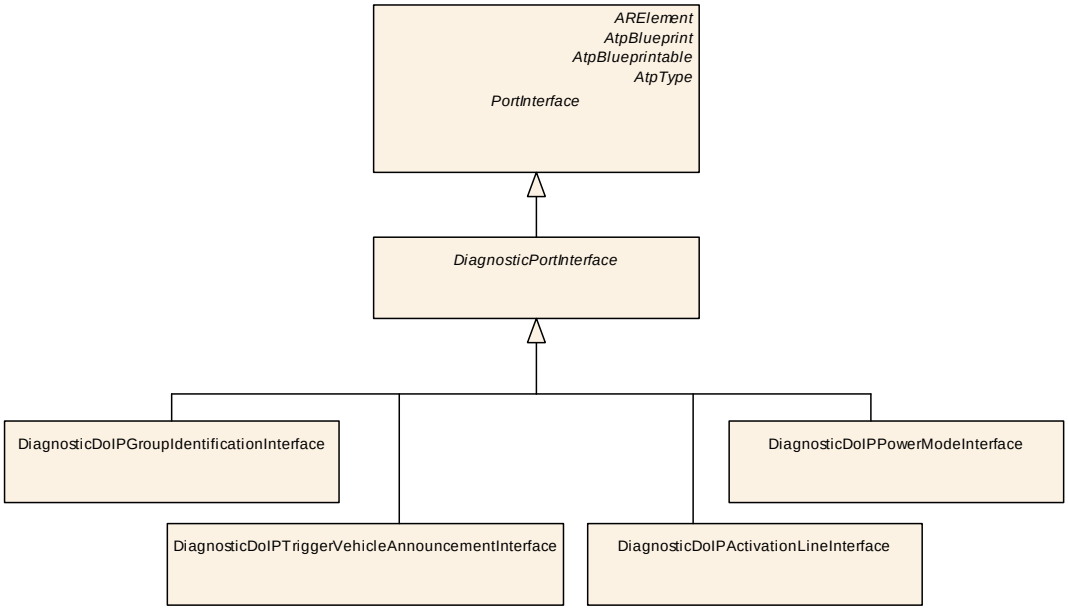


Figure 3.55: Modeling of DoIP `DiagnosticPortInterface`s

Class	DiagnosticDoIPPowerModeInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIP Power Mode on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.87: DiagnosticDoIPPowerModeInterface

Class	DiagnosticDoIPActivationLineInterface
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIPActivationLine on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces

▽

△

Class	DiagnosticDoIPActivationLineInterface			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 3.88: DiagnosticDoIPActivationLineInterface

Class	DiagnosticDoIPTriggerVehicleAnnouncementInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIPTriggerVehicle Announcement on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 3.89: DiagnosticDoIPTriggerVehicleAnnouncementInterface

The DiagnosticDoIPGroupIdentificationInterface, DiagnosticDoIP-PowerModeInterface, DiagnosticDoIPActivationLineInterface and DiagnosticDoIPTriggerVehicleAnnouncementInterface do not require any further details in its formalization.

3.11.12 Diagnostic Interfaces for Upload and Download

[TPS_MANI_01265]{DRAFT} **Semantics of DiagnosticDownloadInterface and DiagnosticDownloadInterface** ⌈The AUTOSAR diagnostic communication API also foresees the existence of DiagnosticPortInterfaces to implement upload and download via diagnostic channels.

Specifically, the following concrete sub-classes of DiagnosticPortInterface are defined to support the implementation of upload and download:

- DiagnosticUploadInterface
- DiagnosticDownloadInterface

⌋*(RS_MANI_00061)*

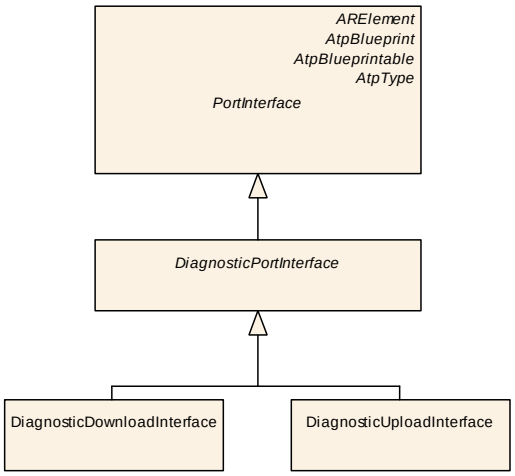


Figure 3.56: Modeling of `DiagnosticUploadInterface` and `DiagnosticDownloadInterface`

Class	**DiagnosticUploadInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for uploading data using diagnostic channels on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.90: DiagnosticUploadInterface

Class	**DiagnosticDownloadInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for downloading data using diagnostic channels on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.91: DiagnosticDownloadInterface

The `DiagnosticUploadInterface` and `DiagnosticDownloadInterface` do not require any further details in its formalization.

3.11.13 Interface to support managing the EcuReset

[TPS_MANI_01332]{DRAFT} **Semantics of `DiagnosticEcuResetInterface`**
⌈Meta-class `DiagnosticEcuResetInterface` represents the ability to support the handling of a request to reset the machine.

This interface will typically be used by the state manager on the AUTOSAR adaptive platform.⌋*(RS_MANI_00061)*

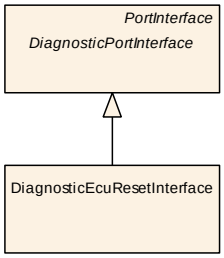


Figure 3.57: Modeling of `DiagnosticEcuResetInterface`

As described in Table 4.21, the creation of a `PortPrototype` typed by a `DiagnosticEcuResetInterface` is done in the context of a `SwcServiceDependency` that aggregates `DiagnosticControlNeeds`.

Class	**DiagnosticEcuResetInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a focused PortInterface for handling the diagnostic service EcuReset on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.92: DiagnosticEcuResetInterface

Class	**DiagnosticControlNeeds**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class indicates a service use-case for reporting the controlled status by diagnostic services.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.93: DiagnosticControlNeeds

3.12 Crypto Interfaces

3.12.1 Interaction with Crypto Software

[TPS_MANI_03253]{DRAFT} **Interaction with crypto software** ⌈Interaction with crypto software on an instance of the *AUTOSAR adaptive application* shall be modeled on the basis of the existence of `RPortPrototype`s typed by a `PortInterface` that is derived from the abstract meta-class `CryptoInterface`.⌋*(RS_MANI_00031)*

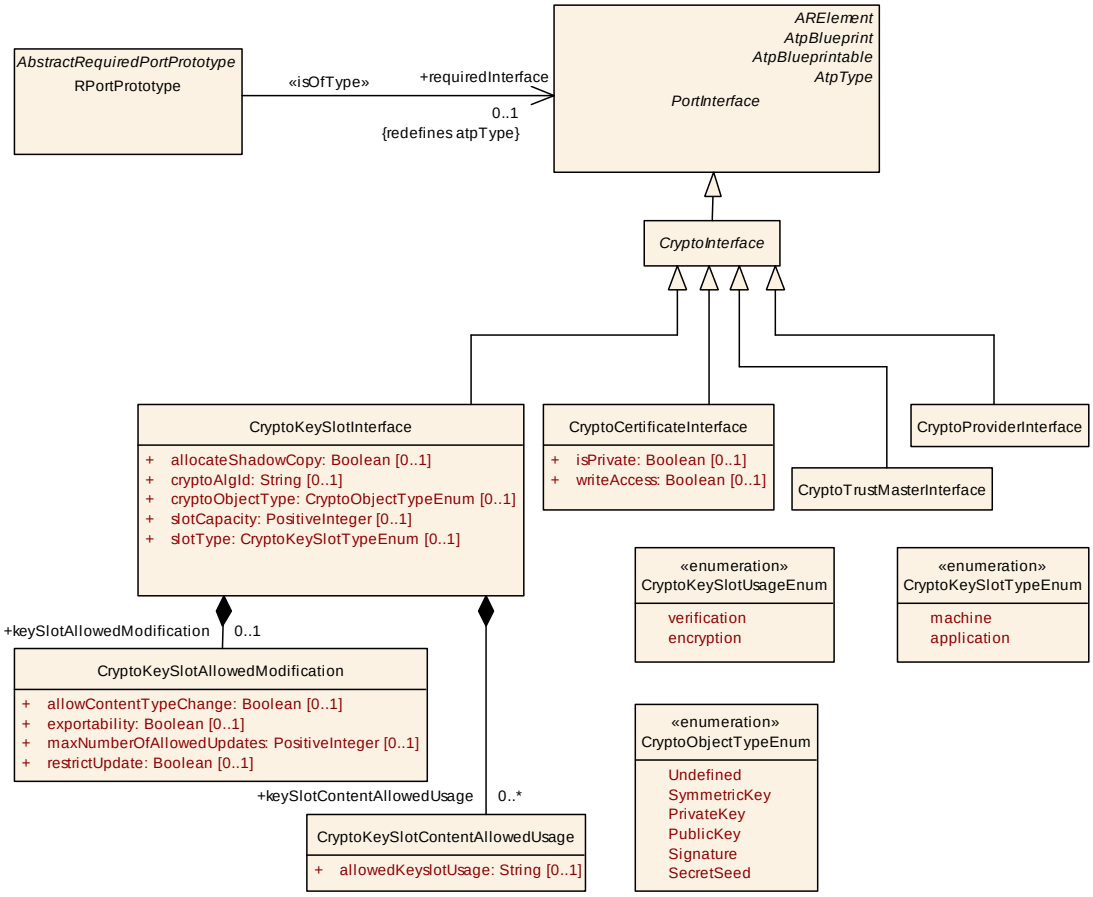


Figure 3.58: `CryptoInterface`s for modeling of the interaction of the Application with the Crypto software

In contrast to the conventions on the AUTOSAR classic Platform, these `CryptoInterface`s are only used on the application side of this communication relation.

The Crypto API is described in [13]. The model-path to an `RPortPrototype` that is referencing a `CryptoInterface` is provided by the `ara::core::InstanceSpecifier` that defines the logical local name used by the application developer in the API call. This local `ara::core::InstanceSpecifier` is translated at runtime with the information from the deployment model to a specific crypto object, e.g. CryptoKeySlot in a CryptoKeyStorage.

Class	**CryptoInterface** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	This meta-class provides the abstract ability to define a PortInterface for the support of crypto use cases. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	CryptoCertificateInterface, CryptoKeySlotInterface, CryptoProviderInterface, CryptoTrustMasterInterface			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.94: CryptoInterface

Figure 3.58 depicts all meta-classes that directly inherit from `CryptoInterface`.

3.12.2 Crypto Key Slot Interface

[TPS_MANI_03254]{DRAFT} **Modeling of application that uses and modifies a Crypto Key** ⌈An Adaptive Application that uses and modifies a Crypto Key is modeled as a `AdaptiveApplicationSwComponentType` with an `RPortPrototype` that is typed by a `CryptoKeySlotInterface` that has the `slotType` value set to `application`.⌋*(RS_MANI_00031)*

[TPS_MANI_03255]{DRAFT} **Modeling of Key Manager application that manages a Crypto Key that is used by Stack Services** ⌈An Key Manager Application that manages a Crypto Key that is used by Stack Services like COM, Persistency or Diagnostic is modeled as a `AdaptiveApplicationSwComponentType` with an `RPortPrototype` that is typed by a `CryptoKeySlotInterface` that has the `slotType` value set to `machine`.⌋*(RS_MANI_00031)*

Class	**CryptoKeySlotInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	This meta-class provides the ability to define a PortInterface for Crypto Key Slots. **Tags:** atp.Status=draft atp.recommendedPackage=CryptoInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
allocateShadowCopy	Boolean	0..1	attr	This attribute defines whether a shadow copy of this Key Slot shall be allocated to enable rollback of a failed Key Slot update campaign (see interface BeginTransaction).

▽

△

Class	**CryptoKeySlotInterface**			
cryptoAlgId	String	0..1	attr	This attribute defines a crypto algorithm restriction (kAlgId Any means without restriction). The algorithm can be specified partially: family & length, mode, padding. Future Crypto Providers can support some crypto algorithms that are not well known/ standardized today, therefore AUTOSAR doesn't provide a concrete list of crypto algorithms' identifiers and doesn't suppose usage of numerical identifiers. Instead of this a provider supplier should provide string names of supported algorithms in accompanying documentation. The name of a crypto algorithm shall follow the rules defined in the specification of cryptography for Adaptive Platform.
cryptoObject Type	CryptoObjectTypeEnum	0..1	attr	Object type that can be stored in the slot. If this field contains "Undefined" then mSlotCapacity must be provided and larger then 0
keySlotAllowed Modification	CryptoKeySlotAllowed Modification	0..1	aggr	Restricts how this keySlot may be used **Tags:**atp.Status=draft
keySlotContent AllowedUsage	CryptoKeySlotContent AllowedUsage	*	aggr	Restriction of allowed usage of a key stored to the slot. **Tags:**atp.Status=draft
slotCapacity	PositiveInteger	0..1	attr	Capacity of the slot in bytes to be reserved by the stack vendor. One use case is to define this value in case that the cryptoObjectType is undefined and the slot size can not be deduced from cryptoObjectType and cryptoAlgId. "0" means slot size can be deduced from cryptoObject Type and cryptoAlgId.
slotType	CryptoKeySlotType Enum	0..1	attr	This attribute defines whether the keySlot is exclusively used by the Application; or whether it is used by Stack Services and managed by a Key Manager Application.

Table 3.95: CryptoKeySlotInterface

Enumeration	**CryptoKeySlotTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign
Note	This enumeration defines the options for the usage of a Key Slot in the platform. **Tags:**atp.Status=draft
Literal	*Description*
application	KeySlot is used and modified exclusively by the Application. **Tags:**atp.EnumerationLiteralIndex=1
machine	Key slot is used by platform modules only. The application manages the key but is not able to use the key. **Tags:**atp.EnumerationLiteralIndex=0

Table 3.96: CryptoKeySlotTypeEnum

Enumeration	**CryptoObjectTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign
Note	Enumeration of all types of crypto objects, i.e. types of content that can be stored to a key slot. **Tags:**atp.Status=draft

▽

△

Enumeration	**CryptoObjectTypeEnum**
Literal	***Description***
PrivateKey	cryp::PrivateKey object **Tags:**atp.EnumerationLiteralIndex=2
PublicKey	cryp::PublicKey object **Tags:**atp.EnumerationLiteralIndex=3
SecretSeed	cryp::SecretSeed object **Tags:**atp.EnumerationLiteralIndex=5
Signature	cryp::Signature object (asymmetric digital signature or symmetric MAC/HMAC) **Tags:**atp.EnumerationLiteralIndex=4
SymmetricKey	cryp::SymmetricKey object **Tags:**atp.EnumerationLiteralIndex=1
Undefined	Object type unknown **Tags:**atp.EnumerationLiteralIndex=0

Table 3.97: CryptoObjectTypeEnum

Please note that the assignment of a `CryptoKeySlot` to a `CryptoProvider` is described in the deployment model (Machine Manifest). With this mapping also the assignment of the `CryptoKeySlot` to a `CryptoPrimitive` of a `CryptoProvider` is established.

But the application developer is able to restrict the usage of the `CryptoKeySlot` to a specific cryptographic algorithm with the attribute `cryptoAlgId`.

To support crypto algorithms that are not well known/ standardized today, AUTOSAR doesn't provide a concrete list of crypto algorithm's identifiers and doesn't suppose usage of numerical identifiers.

Instead of this a provider supplier should provide string names of supported algorithms in accompanying documentation. The name of a crypto algorithm shall follow the rules defined in the specification of cryptography for Adaptive Platform.

In addition the application developer is able to define further requirements for the usage of the `CryptoKeySlot`. With the attribute `cryptoObjectType` the crypto objects that are allowed to be stored in the key slot can be specified.

The allowed modifications of the key slot can be specified by `keySlotAllowedModification`. The allowed usage of the key slot content can be specified by `keySlotContentAllowedUsage`.

The Integrator needs to take the defined settings in the Application Design model into account if the assignment to the `CryptoKeySlot` in the Crypto Storage is performed. Please note that the Application Design model settings are transferred into the deployment model and are therefore are also available at run-time as described in chapter 9.10.2.

Class	**CryptoKeySlotAllowedModification**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	This meta-class restricts the allowed modification of a key stored in the key slot. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allowContent TypeChange	Boolean	0..1	attr	This attribute describes whether the key content type can be changed (true) or not (false), e.g. changing the key from symmetric to RSA.
exportability	Boolean	0..1	attr	This attribute describes whether the key slot content is allowed to be exported or not.
maxNumberOf AllowedUpdates	PositiveInteger	0..1	attr	This attribute describes the maximum updates that are allowed to the slot.
restrictUpdate	Boolean	0..1	attr	This attribute defines whether restrictions on the number of updates are defined or not. False: no restriction is placed on the number of updates. True: restrictions are placed on the number of updates with the attribute maxNumberOfAllowedUpdates.

Table 3.98: CryptoKeySlotAllowedModification

Class	**CryptoKeySlotContentAllowedUsage**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	This meta-class restricts the allowed usage of a key stored in the key slot. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allowedKeyslot Usage	String	0..1	attr	This attribute defines for which operations the KeySlot may be used.

Table 3.99: CryptoKeySlotContentAllowedUsage

[constr_5238]{DRAFT} **CryptoKeySlotAllowedModification.restrictUpdate and the relationship to maxNumberOfAllowedUpdates** ⌈If the CryptoKeySlotAllowedModification.restrictUpdate is set to true then CryptoKeySlotAllowedModification.maxNumberOfAllowedUpdates shall be set to a value.⌋*()*

[constr_5239]{DRAFT} **Predefined values for CryptoKeySlotContentAllowedUsage.allowedKeyslotUsage** ⌈The following values for CryptoKeySlotContentAllowedUsage.allowedKeyslotUsage are predefined by AUTOSAR:

- ALLOW-DATA-ENCRYPTION,
- ALLOW-DATA-DECRYPTION,
- ALLOW-SIGNATURE,
- ALLOW-VERIFICATION,
- ALLOW-KEY-AGREEMENT,
- ALLOW-KEY-DIVERSIFY,

- ALLOW-DRNG-INIT,
- ALLOW-KDF-MATERIAL,
- ALLOW-KEY-EXPORTING,
- ALLOW-KEY-IMPORTING,
- ALLOW-EXACT-MODE-ONLY,
- ALLOW-DERIVED-DATA-ENCRYPTION,
- ALLOW-DERIVED-DATA-DECRYPTION,
- ALLOW-DERIVED-SIGNATURE,
- ALLOW-DERIVED-VERIFICATION,
- ALLOW-DERIVED-DIVERSIFY,
- ALLOW-DERIVED-DRNG-INIT,
- ALLOW-DERIVED-KDF-MATERIAL,
- ALLOW-DERIVED-KEY-EXPORTING,
- ALLOW-DERIVED-KEY-IMPORTING,
- ALLOW-DERIVED-EXACT-MODE-ONLY

⌋()

3.12.3 Crypto Certificate Interface

[TPS_MANI_03256]{DRAFT} **Modeling of application that accesses a Crypto Certificate** ⌈An Adaptive Application that accesses a Crypto Certificate is modeled as a `AdaptiveApplicationSwComponentType` with an `RPortPrototype` that is typed by a `CryptoCertificateInterface`.⌋*(RS_MANI_00031)*

Class	**CryptoCertificateInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	This meta-class provides the ability to define a PortInterface for a CryptoCertificate. **Tags:** atp.Status=draft atp.recommendedPackage=CryptoInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
isPrivate	Boolean	0..1	attr	This attribute controls the possibility to access the content of the CryptoCertificateSlot by Find() interfaces of the X509 Provider.

▽

△

Class	CryptoCertificateInterface			
writeAccess	Boolean	0..1	attr	This attribute defines whether the application has write-access to the CryptoCertificate (True) or only read-access (False).

Table 3.100: CryptoCertificateInterface

3.12.4 Crypto Provider Interface

[TPS_MANI_03257]{DRAFT} **Modeling of application that accesses a Crypto Provider** ⌈An Adaptive Application that accesses a Crypto Provider is modeled as a `AdaptiveApplicationSwComponentType` with an `RPortPrototype` that is typed by a `CryptoProviderInterface`.⌋*(RS_MANI_00031)*

Please note that the `CryptoProviderInterface` shall be used if the Application needs to access a Crypto Provider to execute keyless operations, e.g. Hashing, Random Number Generation. For cryptographic transformations that require keys the `CryptoKeySlotInterface` may be used.

Class	CryptoProviderInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	This meta-class provides the ability to define a PortInterface for a CryptoProvider. **Tags:** atp.Status=draft atp.recommendedPackage=CryptoInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.101: CryptoProviderInterface

3.12.5 Crypto TrustMaster Interface

[TPS_MANI_03258]{DRAFT} **Modeling of application designed as trust-master** ⌈An Adaptive Application designed as trust-master is modeled as a `AdaptiveApplicationSwComponentType` with an `RPortPrototype` that is typed by a `CryptoTrustMasterInterface`.⌋*(RS_MANI_00031)*

An Application requires TrustMaster privileges to set global (machine-wide) root-of-trust certificates. Note: such a certificate may not be private.

Class	**CryptoTrustMasterInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	This meta-class provides the ability to define a PortInterface for TrustMaster. **Tags:** atp.Status=draft atp.recommendedPackage=CryptoInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.102: CryptoTrustMasterInterface

3.12.6 Linking of Crypto Certificate to a Crypto Key Slot

It is possible to model a link between a Crypto Certificate and a Crypto KeySlot in the Application Design with the meta-class `SwcServiceDependency` that aggregates `CryptoCertificateKeySlotNeeds` in the role `serviceNeeds` and `RoleBasedPortAssignment`s that refer to an `RPortPrototype` that is typed by a `CryptoCertificateInterface` and an `RPortPrototype` that is typed by a `CryptoKeySlotInterface`.

[TPS_MANI_03259]{DRAFT} **Linking of Crypto Certificate to a Crypto Key Slot** ⌈

ServiceNeeds kind `CryptoCertificateKeySlotNeeds`

RoleBasedPortAssignment valid roles:

- `CryptoKeySlotInterface` [1]
- `CryptoCertificateInterface` [1]

RoleBasedDataAssignment
N/A

RepresentedPortGroups
N/A

⌋(*RS_MANI_00031*)

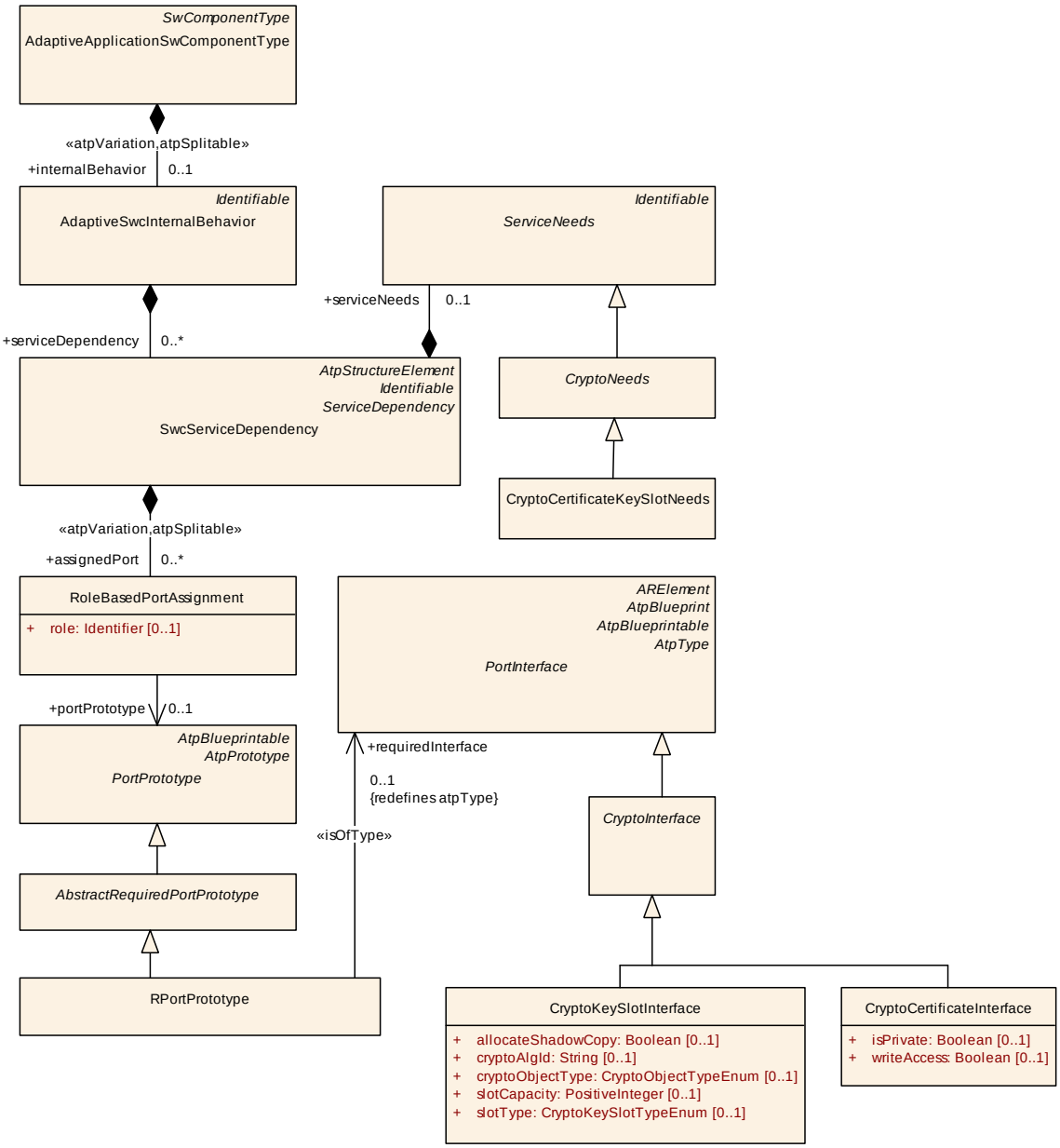


Figure 3.59: Linking of Crypto Certificate with Crypto Key Slot in Application Design

Class	***CryptoNeeds*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	Specifies the abstract needs on the configuration of Crypto. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds*			
Subclasses	CryptoCertificateKeySlotNeeds			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.103: CryptoNeeds

Class	**CryptoCertificateKeySlotNeeds**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	This meta-class shall be taken to indicate that the SwcServiceDependecy modeled with this kind of ServiceNeeds defines a relationship between a CryptoKeySlot and a CryptoCertificate. **Tags:**atp.Status=draft			
Base	*ARObject*, *CryptoNeeds*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.104: CryptoCertificateKeySlotNeeds

The following figure 3.60 shows an example how the `SwcServiceDependency` is used to create a relation between a Crypto Certificate and a Crypto KeySlot.

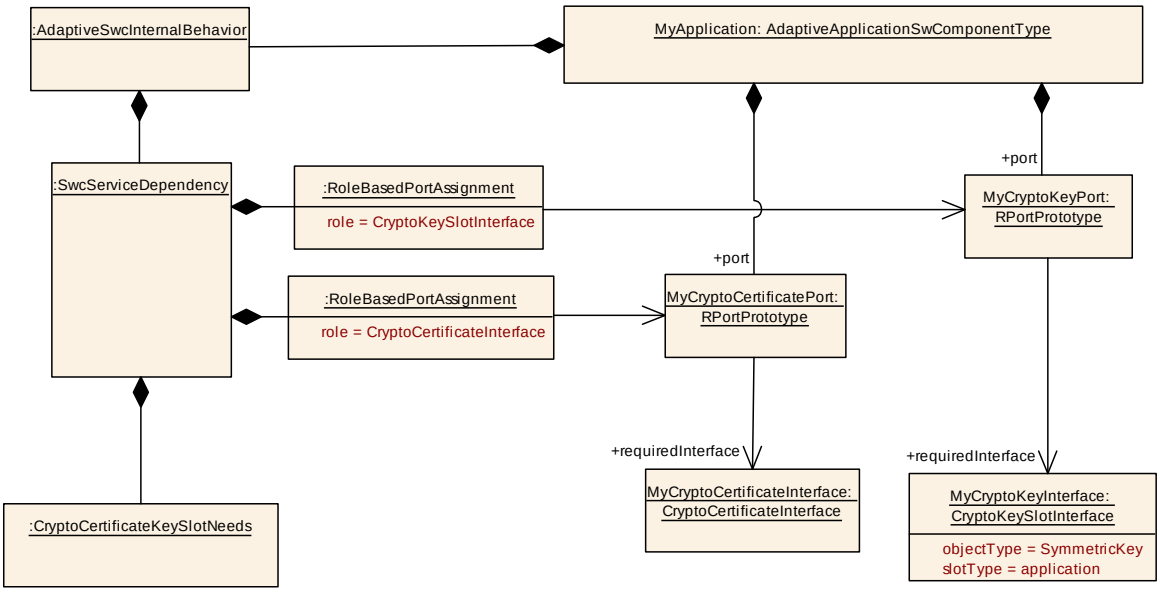


Figure 3.60: Example that shows a link between a Port typed by `CryptoKeySlotInterface` and a Port typed by `CryptoCertificateInterface`

3.13 Raw Data Stream Interface

In some cases it is necessary for the application software to be able to process raw binary data streams sent over a communication channel. Obviously, SOME/IP serialization does not make sense in such a scenario, as would the modeling of `AutosarDataType`s, i.e. the creation of a `ServiceInterface`.

Therefore, a different mechanism that actively supports the requirements of raw data streaming is available on the *AUTOSAR adaptive platform*.

As far as the application software is concerned, the interaction with a raw data stream is based on the usage of an `RPortPrototype` typed by either a `RawDataStreamClientInterface` or a `RawDataStreamServerInterface`.

This kind of PortInterface does neither support nor require any elements with a modeled data type, i.e. an AutosarDataType.

Class	***AbstractRawDataStreamInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class serves as an abstract base class for PortInterfaces related to raw data streams. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	RawDataStreamClientInterface, RawDataStreamServerInterface			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.105: AbstractRawDataStreamInterface

Class	**RawDataStreamClientInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the necessary capabilities for raw data streaming on the client side, i.e. the streaming of data that do not undergo any serialization. Each RawDataStreamClientInterface supports the following capabilities without further modeling: • connect: set up the communication channel • shutdown: close the communication channel • write: send data down the communication channel • read: access incoming data on the communication channel **Tags:** atp.Status=draft atp.recommendedPackage=RawDataStreamInterfaces			
Base	*ARElement*, *ARObject*, *AbstractRawDataStreamInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.106: RawDataStreamClientInterface

Class	**RawDataStreamServerInterface**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface
Note	This meta-class represents the necessary capabilities for raw data streaming on the server side, i.e. the streaming of data that do not undergo any serialization. Each RawDataStreamServerInterface supports the following capabilities without further modeling: • waitForConnection: wait until a communication channel is set up. • shutdown: close the communication channel • write: send data down the communication channel • read: access incoming data on the communication channel **Tags:** atp.Status=draft atp.recommendedPackage=RawDataStreamInterfaces

▽

△

Class	RawDataStreamServerInterface			
Base	ARElement, ARObject, AbstractRawDataStreamInterface, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.107: RawDataStreamServerInterface

3.14 Security Event Report Interface

On the *AUTOSAR adaptive platform*, a dedicated `PortInterface` for the interaction of application-layer software with the AUTOSAR `Intrusion Detection System Manager` ist defined.

The name of this sub-class of abstract meta-class `PortInterface` is `SecurityEventReportInterface`.

Class	SecurityEventReportInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to define a PortInterface for the reporting of security events in the context of the intrusion detection system. **Tags:** atp.Status=draft atp.recommendedPackage=SecurityEventReportInterfaces			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.108: SecurityEventReportInterface

[TPS_MANI_01340]{DRAFT} **Semantics of `SecurityEventReportInterface`** ⌈Each `RPortPrototype` typed by a `SecurityEventReportInterface` is able to report exactly one security event.⌋*()*

[TPS_MANI_01338]{DRAFT} **Semantics of `SecurityEventReportToSecurityEventDefinitionMapping`** ⌈The modeling of the association between a specific security event and the corresponding `RPortPrototype` typed by a `SecurityEventReportInterface` is created by means of the `SecurityEventReportToSecurityEventDefinitionMapping`.⌋*()*

Class	**SecurityEventReportToSecurityEventDefinitionMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the ability to map a PortPrototype for reporting a security event to the actual security event that shall be reported by this PortPrototype. **Tags:** atp.Status=draft atp.recommendedPackage=SecurityEventReportToSecurityEventDefinitionMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
reported SecurityEvent	AbstractRequiredPort Prototype	0..1	iref	This identifies the mapped security event. **Tags:**atp.Status=draft **InstanceRef implemented by:**RPortInComposition InstanceRef
securityEvent Definition	SecurityEventDefinition	0..1	ref	This reference identifies the definition of the security event. **Tags:**atp.Status=draft

Table 3.109: SecurityEventReportToSecurityEventDefinitionMapping

This meta-class maps the `RPortPrototype` to a `SecurityEventDefinition` that itself is part of the so-called `Security Extract`.

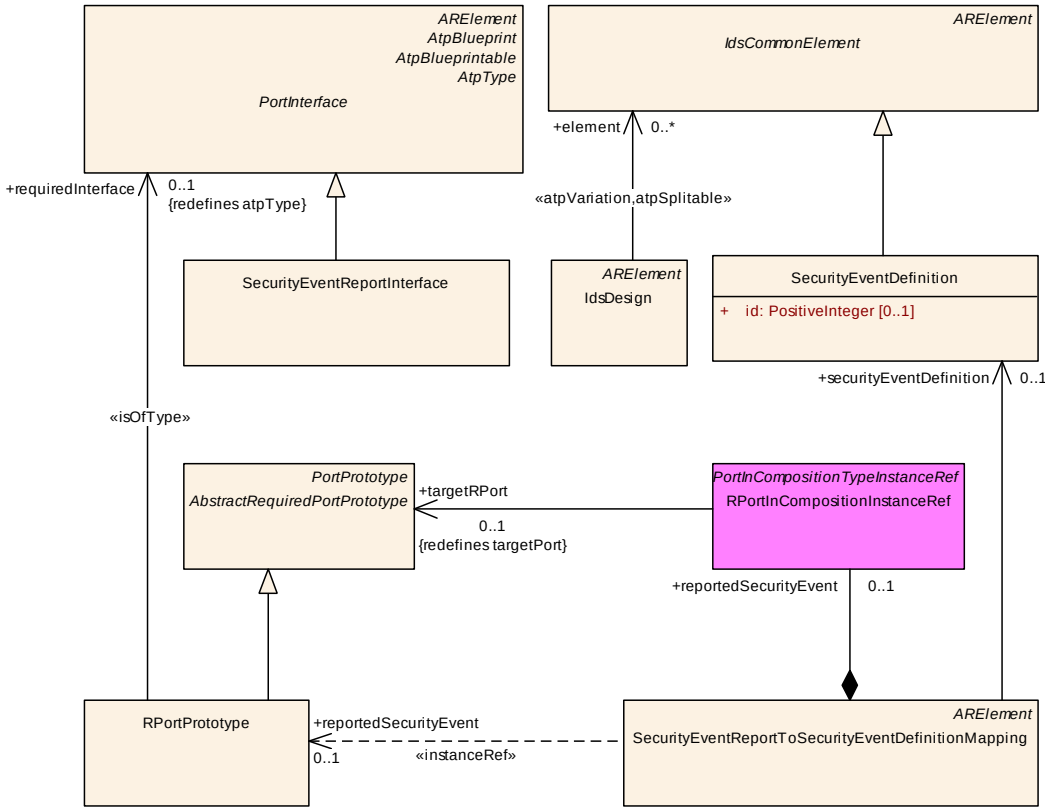


Figure 3.61: Specification of the `SecurityEventReportInterface` and `SecurityEventReportToSecurityEventDefinitionMapping`

[TPS_MANI_01339]{DRAFT} **Existence of the `SecurityEventReportToSecurityEventDefinitionMapping` is motivated by the AUTOSAR methodology**

⌈The existence of the SecurityEventReportToSecurityEventDefinitionMapping is motivated by the AUTOSAR methodology. At the point in time when a given SecurityEventReportInterface is defined it could be that the corresponding SecurityEventDefinition is not yet defined.

So it is possible to add this association later. Another reason for the existence of the mapping class is that a specific piece of application software may report different specific security events defined by different OEMs, depending on the deployment of the application software.

Of course, the semantics of the security event all always be either identical or at least comparable, it could still happen that the Id of a security event might change depending on the specific project or simply because different OEMs use different Ids for semantically identical security events.⌋*()*

3.15 Interaction Endpoint for Application

The interaction of software-components with the outside world can take several forms, e.g. service-oriented communication or the interaction with a persistent data storage.

A formal representation of the interaction needs to be described as an anchor point for adding various additional configuration attributes that make sense in this context but would not make sense in the context of a PortInterface.

There is a model element that already has a long-standing tradition in the AUTOSAR meta-model for exactly the described purpose: the PortPrototype.

The following sub-chapters discuss the interaction by means of PortPrototypes with software "outside" a given software-component with the focus on different kinds of interaction that require different ways to further contribute model elements for configuration.

3.15.1 Service-oriented Communication

The service-oriented communication by means of PortPrototypes does **not** support the concept of a communication endpoint that is both required and provided **at the same time**. This motivates the existence of [constr_1473].

[constr_1473]{DRAFT} **No support for PRPortPrototype** ⌈A ServiceInterface shall not be referenced by a PRPortPrototype in the role providedRequiredInterface.⌋*()*

[TPS_MANI_01039]{DRAFT} **Representation of provided service** ⌈A **provided service** shall be modeled by means of an PPortPrototype that is typed by a ServiceInterface.⌋*(RS_MANI_00002)*

[TPS_MANI_01040]{DRAFT} **Representation of required service** ⌈A **required service** shall be modeled by means of an RPortPrototype that is typed by a ServiceInterface.⌋*(RS_MANI_00002)*

For more background regarding the rationale of [constr_1473], please refer to [1].

Please note that the utilization of service discovery on the *AUTOSAR adaptive platform* means that opposite communication ends **are by design not known upfront**.

As a consequence, it is in general not possible to use AssemblySwConnectors to model a pre-defined relation between two communication endpoints modeled as PortPrototypes.

Independent of the issue described above, it is still necessary to provide means for configuration of a given PortPrototype on different levels:

- The PortPrototype itself (i.e. as a whole) may need to be customized, independently of the kind or number of elements aggregated by the corresponding ServiceInterface. This aspect is discussed in section 3.15.4.
- The usage of elements of the corresponding ServiceInterface may need to be configured for a given PortPrototype. This aspect is discussed in section 3.15.5.

3.15.2 Interaction with Persistent Key-Value Storage

The usage of PortPrototypes for the purpose of interacting with *persistent key-value storage* is less restricted than in the case of service-oriented communication. In other words, it is perfectly valid to use a PRPortPrototype where applicable.

[TPS_MANI_01073]{DRAFT} **Semantics of PortPrototype typed by PersistencyKeyValueStorageInterface** ⌈The usage of a specific sub-class of PortPrototype typed by PersistencyKeyValueStorageInterface indicates the intended semantics of interaction:

- The usage of a RPortPrototype indicates that the persistent data can only be **read from** the persistent storage.
- The usage of a PPortPrototype indicates that the persistent data can only be **written to** the persistent storage.
- The usage of a PRPortPrototype indicates that the persistent data can be **read from** as well as **written to** the persistent storage.

⌋*(RS_MANI_00027)*

Please note that the PersistencyKeyValueStorageInterface is described in chapter 3.8.2.

3.15.3 Interaction with Persistent File Storage

Interaction with **persistent file storage** can involve the ability to read from and write to a file by the same application. Therefore, the existence of a PRPortPrototype typed by a PersistencyFileStorageInterface shall be supported.

[TPS_MANI_01081]{DRAFT} **Semantics of PortPrototype typed by PersistencyFileStorageInterface** ⌈The usage of a specific sub-class of PortPrototype typed by PersistencyFileStorageInterface indicates the intended semantics of interaction:

- The usage of a RPortPrototype indicates that the corresponding file(s) can be **opened for read access**.
- The usage of a PPortPrototype indicates that the corresponding file(s) can be **opened or created for write access**. Also, there is the ability to **delete** a file.
- The usage of a PRPortPrototype indicates that the corresponding file(s) can be **opened or created for read and write access**. Also, there is the ability to **delete** a file.

⌋*(RS_MANI_00027)*

Please note that the PersistencyFileStorageInterface is described in chapter 3.8.3.

3.15.4 Port Prototype Props

As mentioned before, in some cases a qualification of the semantics of PortPrototypes is necessary. For this purpose, AUTOSAR typically defines a *props* class of some kind. The same approach applies in this situation as well.

In particular, PortPrototype aggregates the abstract meta-class PortPrototypeProps, that in turn starts an inheritance tree of derived meta-classes that have the ability to qualify sub-classes of PortPrototype accordingly.

One example for this approach is the definition of the meta-class RPortPrototypeProps, sketched in Figure 3.62.

[constr_3359]{DRAFT} **RPortPrototypeProps are related only to RPortPrototypes** ⌈The RPortPrototypeProps shall be aggregated only by a RPortPrototype in the role portPrototypeProps.⌋*()*

[TPS_MANI_01057]{DRAFT} **Semantics of RPortPrototypeProps.searchIntention** ⌈The value of the attribute RPortPrototypeProps.searchIntention clarifies whether the search for a corresponding offer shall be done as a search for all or else as a search for a specific ID.

Typically, a search for any results in a collection of offers while the search for a given id results in just a single offer.⌋*(RS_MANI_00002)*

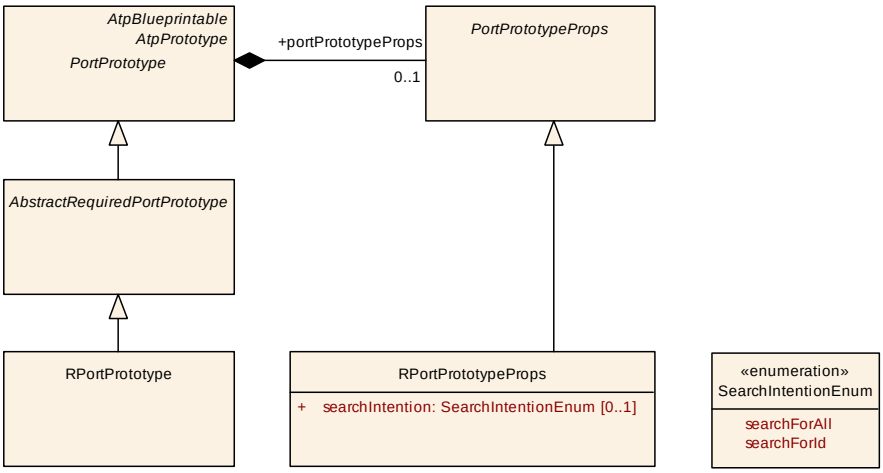


Figure 3.62: Modeling of the `RPortPrototypeProps` for `RPortPrototype`

Class	***PortPrototypeProps*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This meta-class represents the ability to define a further qualification of semantics of sub-classes of Port Prototype. **Tags:**atp.Status=draft			
Base	*ARObject*			
Subclasses	RPortPrototypeProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.110: PortPrototypeProps

Class	**RPortPrototypeProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	PortPrototypeProps for a RPort. **Tags:**atp.Status=draft			
Base	*ARObject*, *PortPrototypeProps*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
searchIntention	SearchIntentionEnum	0..1	attr	This attribute is used to specify the intention of the developer of the enclosing software-component in terms of whether the respective PortPrototype shall be use to search for a specific service instance or all instances of the given service. Please note that the value of this attribute does not create a binding contract. The actual search behavior is defined as part of the service instance manifest.

Table 3.111: RPortPrototypeProps

Enumeration	**SearchIntentionEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign
Note	This meta-class allows for the definition of a dedicated search intention from the application's point of view. **Tags:**atp.Status=draft
Literal	***Description***
searchForAll	This value represents the intention to search for all instances of the given service **Tags:**atp.EnumerationLiteralIndex=0
searchForId	This value represents the intention to search for a dedicated instance of the given service. **Tags:**atp.EnumerationLiteralIndex=1

Table 3.112: SearchIntentionEnum

3.15.5 Port Prototype ComSpec

[TPS_MANI_01053]{DRAFT} **Usage of `ComSpec`s on the *AUTOSAR adaptive platform*** ⌈The aspect of further qualification of elements of the `ServiceInterface` used to type given `PortPrototype` is implemented by means of `ComSpec`s, i.e. specific sub-classes of the abstract meta-classes `RPortComSpec` and `PPortComSpec`.

However, the support for `ComSpec`s on the *AUTOSAR adaptive platform* only covers a **limited selection** of attributes of a specific `ComSpec`.⌋*(RS_MANI_00002)*

The details about supported attributes of either a `RPortComSpec` or `PPortComSpec` are described in this chapter.

The configuration of transformation capabilities in the context of a `ComSpec` is possible by means of subclasses of meta-class `TransformationComSpecProps`.

Class	***TransformationComSpecProps*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	TransformationComSpecProps holds all the attributes for transformers that are port specific.			
Base	*ARObject*, *Describable*			
Subclasses	EndToEndTransformationComSpecProps, UserDefinedTransformationComSpecProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.113: TransformationComSpecProps

Class	**UserDefinedTransformationComSpecProps**
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication
Note	The UserDefinedTransformationComSpecProps is used to specify port specific configuration properties for custom transformers.
Base	*ARObject*, *Describable*, *TransformationComSpecProps*

▽

△

Class	UserDefinedTransformationComSpecProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.114: UserDefinedTransformationComSpecProps

Class	EndToEndTransformationComSpecProps			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	The class EndToEndTransformationIComSpecProps specifies port specific configuration properties for EndToEnd transformer attributes.			
Base	*ARObject*, *Describable*, *TransformationComSpecProps*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
clearFromValid ToInvalid	Boolean	0..1	attr	Clear monitoring window on transition from state Valid to state Invalid.
disableEndTo EndCheck	Boolean	0..1	attr	Disables/Enables the E2E check. The E2Eheader is removed from the payload independent from the setting of this attribute.
disableEndTo EndState Machine	Boolean	0..1	attr	Disables the E2EStateMachine (only E2E check functionality is performed)
e2eProfile Compatibility Props	E2EProfileCompatibility Props	0..1	ref	Reference to additional settings for the E2E state machine.
maxDelta Counter	PositiveInteger	0..1	attr	Maximum allowed difference between two counter values of two consecutively received valid messages. For example, if the receiver gets data with counter 1 and Max DeltaCounter is 3, then at the next reception the receiver can accept Counters with values 2, 3 or 4.
maxErrorState Init	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_INIT. The minimum value is 0.
maxErrorState Invalid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_INVALID. The minimum value is 0.
maxErrorState Valid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_VALID. The minimum value is 0.
minOkStateInit	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INIT. The minimum value is 1.
minOkState Invalid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INVALID. The minimum value is 1.
minOkState Valid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_VALID. The minimum value is 1.

▽

△

Class	EndToEndTransformationComSpecProps			
windowSizeInit	PositiveInteger	0..1	attr	Size of the monitoring window of state Init for the E2E state machine.
windowSize Invalid	PositiveInteger	0..1	attr	Size of the monitoring window of state Invalid for the E2E state machine.
windowSize Valid	PositiveInteger	0..1	attr	Size of the monitoring window of state Valid for the E2E state machine.

Table 3.115: EndToEndTransformationComSpecProps

[TPS_MANI_01327]{DRAFT} **Value of `EndToEndTransformationComSpecProps.disableEndToEndCheck` vs. value of `EndToEndTransformationComSpecProps.disableEndToEndStateMachine`** ⌈If the value of attribute `EndToEndTransformationComSpecProps.disableEndToEndCheck` is set to `True`, then the value of attribute `EndToEndTransformationComSpecProps.disableEndToEndStateMachine` shall be ignored.⌋*(RS_MANI_00028)*

3.15.5.1 Port Prototypes typed by Service Interfaces

3.15.5.1.1 Receiver ComSpec

It is necessary to provide means to configure the queue length of the reception of an `event` on a case-by-case basis. In other words, even two “adjacent” `event`s within the same `RPortPrototype` may need a different handling of the queue length.

[TPS_MANI_01054]{DRAFT} **Definition of the queue length of an `event` or `field` notifier** ⌈The definition of the queue length of an `event` or `field` notifier shall be modeled by means of the attribute `QueuedReceiverComSpec.queueLength`.⌋*(RS_MANI_00026)*

The `ReceiverComSpec` needs an attribute that indicates whether the enclosing `AdaptiveApplicationSwComponentType` has an intention to actually access the referenced `dataElement`. This attribute represents a security feature related to identity and access management [14].

Specifically, this aspect is typically summarized as a capability of the software, i.e. the `AdaptiveApplicationSwComponentType` expresses expresses its capability with respect to the specific `dataElement`. The term "capability" is an integral part of the jargon in the domain of identity and access management.

However, outside the identity and access management domain, this terminology is sometimes hard to motivate. What could be motivated is that the `AdaptiveApplicationSwComponentType` expresses its *intent* to actually access the `dataElement`.

From that perspective, the process of adding an event to a `ServiceInterface` adds the capability to use the `dataElement`. But whether the software that uses the `ServiceInterface` actually intends to access the `dataElement` can be expressed by an attribute in the `ReceiverComSpec` named `receiverIntent`.

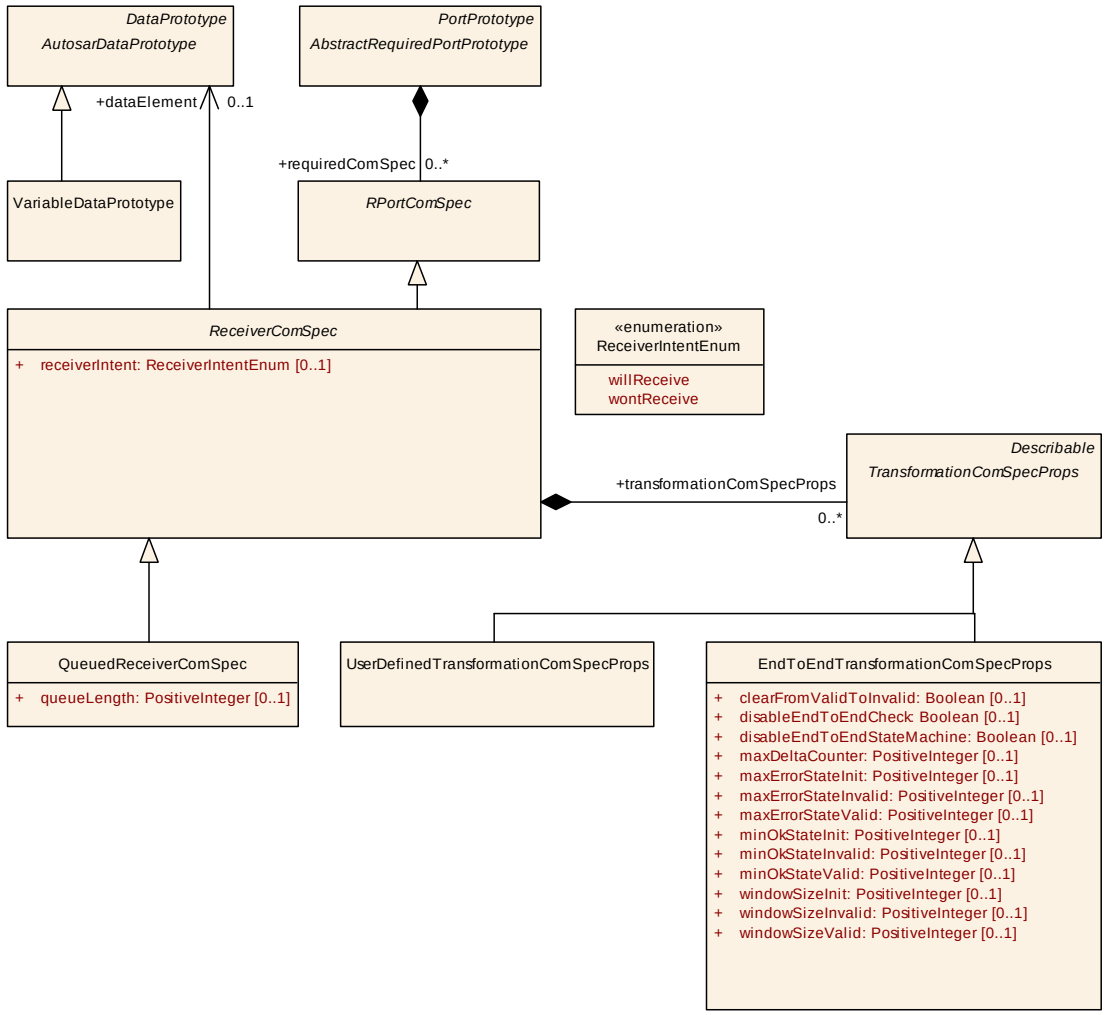


Figure 3.63: Modeling of the `ReceiverComSpec` on the *AUTOSAR adaptive platform*

Class	***ReceiverComSpec*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Receiver-specific communication attributes (RPortPrototype typed by ServiceInterface) that are relevant for events and field notifiers.			
Base	*ARObject*, *RPortComSpec*			
Subclasses	NonqueuedReceiverComSpec, QueuedReceiverComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	AutosarDataPrototype	0..1	ref	Data element these attributes belong to.
receiverIntent	ReceiverIntentEnum	0..1	attr	This attribute represents the expressed intent of the receiver. The receiver may decide to claim that existing resources of a ServiceInterface are expressly not used by this specific receiver. The conceptual background of this claim may be driven by security, safety, etc. **Tags:**atp.Status=draft
receptionProps	ReceptionComSpec Props	0..1	aggr	"This aggregation represents the definition transmission props in the context of the enclosing ReceiverComSpec.

△

Class	***ReceiverComSpec*** (abstract)			
transformation ComSpecProps	TransformationCom SpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.

Table 3.116: ReceiverComSpec

Class	**QueuedReceiverComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes specific to queued receiving.			
Base	*ARObject*, *RPortComSpec*, *ReceiverComSpec*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
queueLength	PositiveInteger	0..1	attr	Length of queue for received events.

Table 3.117: QueuedReceiverComSpec

[TPS_MANI_01106]{DRAFT} **Specification of intentions for the receiver of `events` or `field` notifiers** ⌈The attribute `ReceiverComSpec.receiverIntent` can be used to specify whether the software actually intends to access the referenced `event`s or `field` notifier or whether it explicitly states that it is not interested in the value.⌋*(RS_MANI_00034)*

Enumeration	**ReceiverIntentEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec
Note	This meta-class represents the intent to specify how a given ServiceInterface is used from the perspective of a given event receiver. **Tags:**atp.Status=draft
Literal	***Description***
willReceive	The receiver will receive the event or field notifier. **Tags:**atp.EnumerationLiteralIndex=0
wontReceive	The receiver won't receive the event or field notifier. **Tags:**atp.EnumerationLiteralIndex=1

Table 3.118: ReceiverIntentEnum

[TPS_MANI_03132]{DRAFT} **Semantics of E2E attributes in `ReceiverComSpec`** ⌈The `EndToEndTransformationComSpecProps` shall be used for the specification of `RPortPrototype`-specific configuration options related to end-to-end protection of events or field notifiers.⌋*(RS_MANI_00028)*

3.15.5.1.2 Sender ComSpec

The `SenderComSpec` is modeled in the same way as described in the Software Component Template [1].

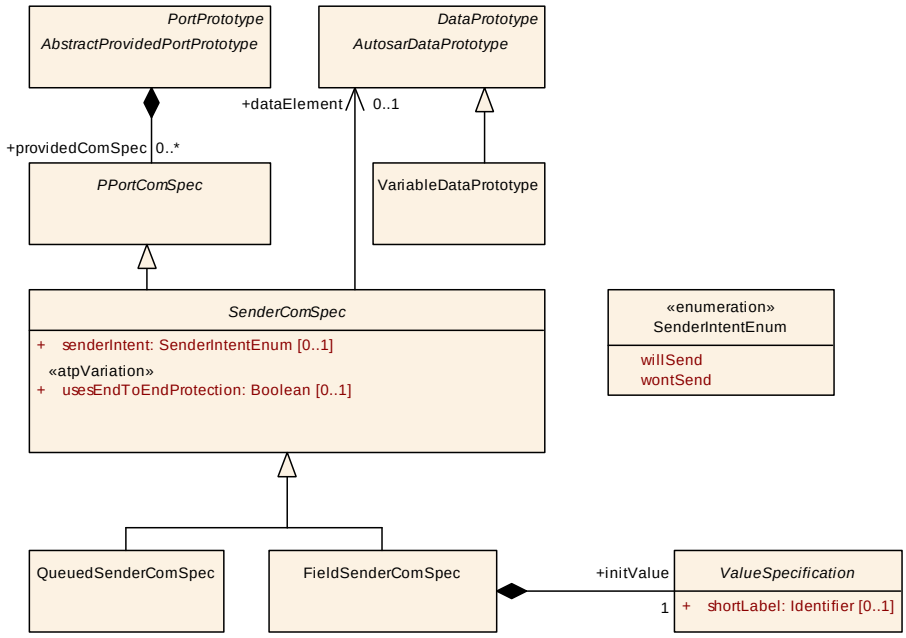


Figure 3.64: Modeling of the `SenderComSpec` on the *AUTOSAR adaptive platform*

Class	***SenderComSpec*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes for a sender port (PPortPrototype typed by ServiceInterface) that are relevant for events and field notifiers.			
Base	*ARObject*, *PPortComSpec*			
Subclasses	FieldSenderComSpec, NonqueuedSenderComSpec, QueuedSenderComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	AutosarDataPrototype	0..1	ref	Data element these quality of service attributes apply to.
senderIntent	SenderIntentEnum	0..1	attr	This attribute represents the expressed intent of the client. The client may decide to claim that existing resources of a ServiceInterface are expressly not used by this specific client. The conceptual background of this claim may be driven by security, safety, etc. **Tags:**atp.Status=draft
transmission Props	TransmissionComSpec Props	0..1	aggr	This aggregation represents the definition transmission props in the context of the enclosing SenderComSpec.
usesEndToEnd Protection	Boolean	0..1	attr	This indicates whether the corresponding dataElement shall be transmitted using end-to-end protection. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime

Table 3.119: SenderComSpec

[TPS_MANI_03210]{DRAFT} **Specification of `event` specific communication attributes** ⌈The meta-class `QueuedSenderComSpec` can be used to specify communication attributes that are relevant for an `event` on the sender side.⌋ *(RS_MANI_00002)*

Class	QueuedSenderComSpec			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes specific to distribution of events (PPortPrototype, SenderReceiverInterface and dataElement carries an "event").			
Base	*ARObject*, *PPortComSpec*, *SenderComSpec*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.120: QueuedSenderComSpec

[TPS_MANI_03211]{DRAFT} **Specification of `field` specific communication attributes** ⌈The meta-class `FieldSenderComSpec` can be used to specify communication attributes that are relevant for a `field` on the sender side.⌋ *(RS_MANI_00002)*

Class	FieldSenderComSpec			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec			
Note	Port specific communication attributes for a Field that is defined in a ServiceInterface. **Tags:**atp.Status=draft			
Base	*ARObject*, *PPortComSpec*, *SenderComSpec*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
initValue	ValueSpecification	1	aggr	Initial value for a Field that is set before the Service Interface is offered. **Tags:**atp.Status=draft

Table 3.121: FieldSenderComSpec

[TPS_MANI_03212]{DRAFT} **Specification of initial value for a `field`** ⌈The attribute `FieldSenderComSpec.initValue` can be used to specify an initial Value for a `field`.⌋ *(RS_MANI_00002)*

A `field` has a valid value at any time as described in subsection 3.4.3. ara::com ensures that a service implementation providing a field has a field value before the field becomes visible to potential consumers.

This is explained in more detail in [15] where it is defined that the initial field value shall be set at least once via Update() by the application code before `OfferService()` gets called.

Custom-code (e.g. component model above ara::com) may use the defined `initValue` to call `Field.Update(initValue)`.

[TPS_MANI_01107]{DRAFT} **Specification of intentions for the sender of `event`s or `field` notifiers** ⌈The attribute `SenderComSpec.senderIntent` can be used to specify whether the software actually intends to send the referenced `event`s or `field` notifier.⌋ *(RS_MANI_00034)*

Enumeration	**SenderIntentEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec
Note	This meta-class represents the intent to specify how a given ServiceInterface is used from the perspective of a given event sender. **Tags:**atp.Status=draft
Literal	***Description***
willSend	The sender will send the event or field notifier. **Tags:**atp.EnumerationLiteralIndex=0
wontSend	The sender won't send the event or field notifier. **Tags:**atp.EnumerationLiteralIndex=1

Table 3.122: SenderIntentEnum

3.15.5.1.3 Client ComSpec

The `ClientComSpec` undergoes extensions for the *AUTOSAR adaptive platform*, namely the ability to refer to the getter and setter method of a `field` and the definition of intentions.

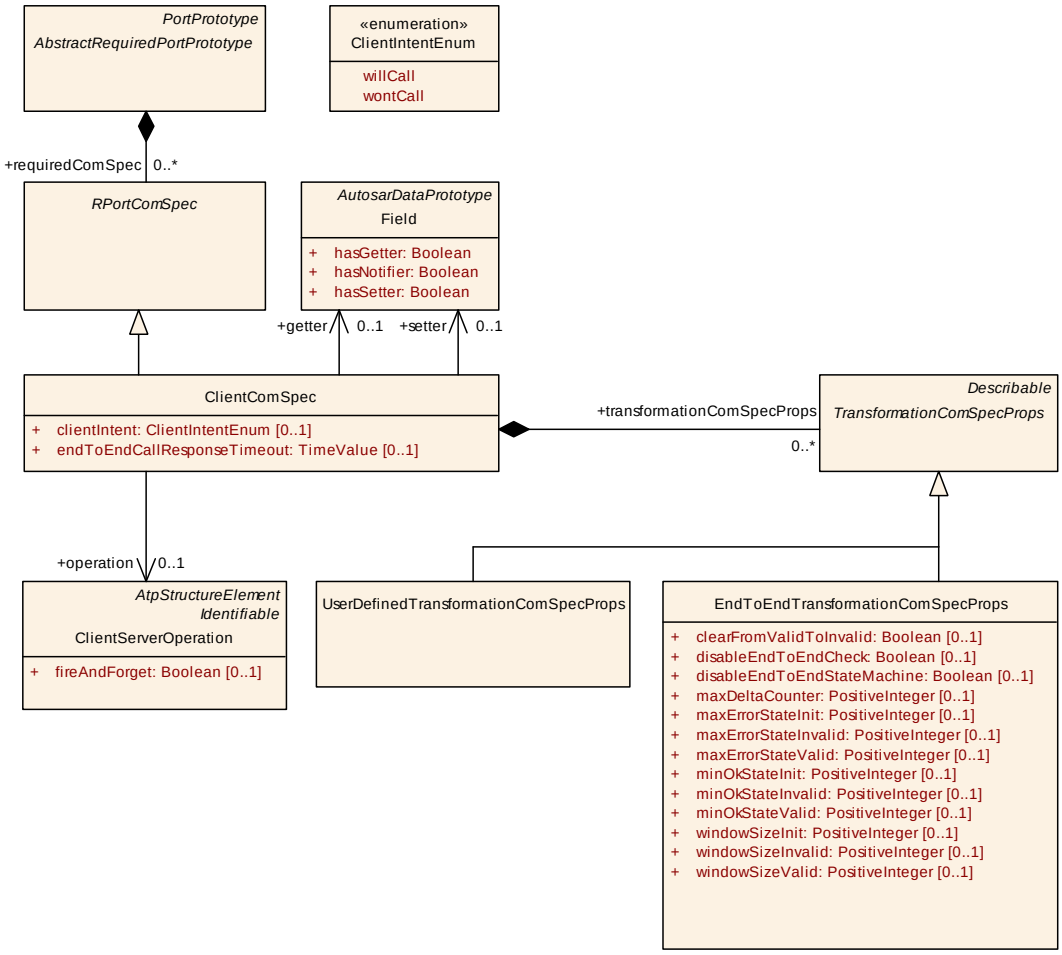


Figure 3.65: Modeling of the `ClientComSpec` on the *AUTOSAR adaptive platform*

Class	**ClientComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Client-specific communication attributes (RPortPrototype typed by ServiceInterface) that are relevant for methods and field getters and setters.			
Base	*ARObject*, *RPortComSpec*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientIntent	ClientIntentEnum	0..1	attr	This attribute represents the expressed intent of the sender. The sender may decide to claim that existing resources of a ServiceInterface are expressly not used by this specific sender. The conceptual background of this claim may be driven by security, safety, etc." **Tags:**atp.Status=draft
endToEndCall Response Timeout	TimeValue	0..1	attr	This attribute defines the maximum time interval in which the application shall expect the servers's response (time between the sending of the call invocation until the arrival of the server's response).
getter	Field	0..1	ref	The existence of this reference indicates that the Client ComSpec refers to the getter of a Field. **Tags:**atp.Status=draft
operation	ClientServerOperation	0..1	ref	This represents the corresponding ClientServerOperation.
setter	Field	0..1	ref	The existence of this reference indicates that the Client ComSpec refers to the setter of a Field. **Tags:**atp.Status=draft
transformation ComSpecProps	TransformationCom SpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.

Table 3.123: ClientComSpec

[TPS_MANI_01108]{DRAFT} **Specification of intentions for the caller of a `methods` or `field` setter/getter** ⌈The attribute `ClientComSpec.clientIntent` can be used to specify whether the software actually intends to call the referenced `methods` or getter/setter of a referenced `field`.⌋*(RS_MANI_00034)*

Enumeration	**ClientIntentEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec
Note	This meta-class represents the intent to specify how a given ServiceInterface is used from the perspective of a given client. **Tags:**atp.Status=draft
Literal	***Description***
willCall	The client will call this method. **Tags:**atp.EnumerationLiteralIndex=0
wontCall	The client won't call this method. **Tags:**atp.EnumerationLiteralIndex=1

Table 3.124: ClientIntentEnum

[TPS_MANI_01324]{DRAFT} **Semantics of E2E attributes in `ClientComSpec`** ⌈The `EndToEndTransformationComSpecProps` shall be used for the specification of `RPortPrototype`-specific configuration options related to end-to-end protection of `methods`.⌋*(RS_MANI_00028)*

3.15.5.1.4 Server ComSpec

The `ServerComSpec` undergoes extensions for the AUTOSAR adaptive platform, namely the ability to refer to the `getter` and `setter` `method` of a `field` and the definition of intentions.

[TPS_MANI_01325]{DRAFT} **Semantics of E2E attributes in `ServerComSpec`** ⌈The `EndToEndTransformationComSpecProps` shall be used for the specification of `PPortPrototype`-specific configuration options related to end-to-end protection of `method`s.⌋ *(RS_MANI_00028)*

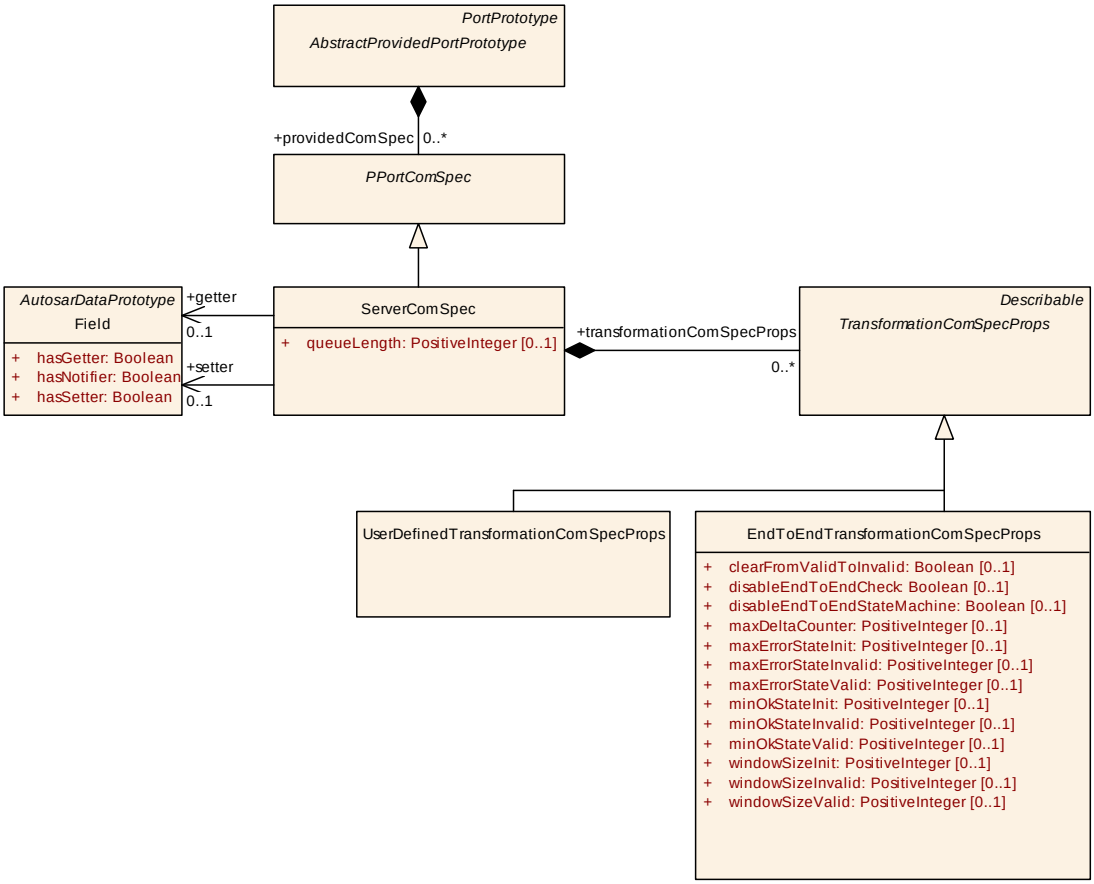


Figure 3.66: Modeling of the `ServerComSpec` on the *AUTOSAR adaptive platform*

Class	**ServerComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Server-specific communication attributes (PPortPrototype typed by ServiceInterface) that are relevant for methods and field getters and setters.			
Base	*ARObject*, *PPortComSpec*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**ServerComSpec**			
getter	Field	0..1	ref	The existence of this reference indicates that the Server ComSpec refers to the getter of a Field. **Tags:**atp.Status=draft
operation	ClientServerOperation	0..1	ref	Operation these communication attributes apply to.
queueLength	PositiveInteger	0..1	attr	Length of call queue on the server side.
setter	Field	0..1	ref	The existence of this reference indicates that the Server ComSpec refers to the setter of a Field. **Tags:**atp.Status=draft
transformation ComSpecProps	TransformationCom SpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.

Table 3.125: ServerComSpec

3.15.5.2 Port Prototypes typed by Persistency Data Interfaces

[TPS_MANI_01314]{DRAFT} **Further qualification of properties of `PortPrototype`s typed by `PersistencyKeyValueStorageInterface`s** ⌈For `PortPrototype`s typed by `PersistencyKeyValueStorageInterface`s it is possible to define further qualifying attributes for the required side.

For this purpose meta-class `PersistencyDataRequiredComSpec` is provided.⌋ *(RS_MANI_00027)*

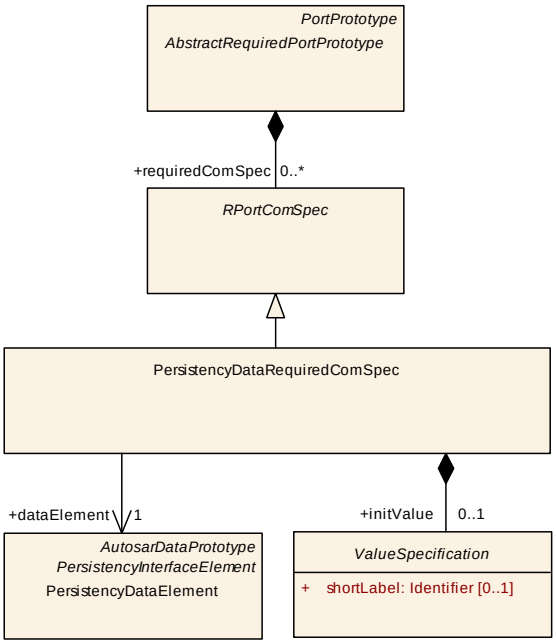


Figure 3.67: Modeling of `ComSpec` for persistency

[TPS_MANI_01160]{DRAFT} **Definition of initial value for `PersistencyDataElement`** ⌈The definition of an initial value for a `PersistencyDataElement`

can be done on the level of a PortPrototype by means of PersistencyDataRequiredComSpec.initValue⌋(*RS_MANI_00027*)

Class	**PersistencyDataRequiredComSpec**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec			
Note	This meta-class represents the ability to define port-specific attributes for supporting use cases of data persistency on the required side. **Tags:**atp.Status=draft			
Base	*ARObject*, *RPortComSpec*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	PersistencyData Element	1	ref	This refrence represents the PersistencyDataElement for which the PersistencyDataRequiredComSpec applies. **Tags:**atp.Status=draft
initValue	ValueSpecification	0..1	aggr	This aggregation represents the definition of an initial value for the PersistencyDataElement referenced by the enclosing PersistencyDataRequiredComSpec **Tags:**atp.Status=draft

Table 3.126: PersistencyDataRequiredComSpec

3.16 Executable

[TPS_MANI_01010]{DRAFT} **Root element for a hierarchical software-component** ⌈Executable aggregates meta-class RootSwComponentPrototype in the role rootSwComponentPrototype to provide a root element for an arbitrarily nested hierarchy of software-components represented by the reference RootSwComponentPrototype.applicationType.⌋(*RS_MANI_00004*)

Please note that the aggregation of RootSwComponentPrototype by Executable is the basis for the applicability of an ≪instanceRef≫ reference into the hierarchy of software-components that represent the functionality of the Executable.

This modeling approach is similar to the modeling of a System on the *AUTOSAR classic platform*.

[TPS_MANI_01279]{DRAFT} **Semantics of Executable.reportingBehavior** ⌈Attribute Executable.reportingBehavior shall be used to control the reporting of the execution state of the enclosing Executable to the Execution Management. If the attribute does not exist, the Executable shall report its execution state to the Execution Management.⌋(*RS_MANI_00023*)

[constr_1605]{DRAFT} **Standardized values of attribute Executable.category** ⌈The following values for attribute Executable.category are standardized by AUTOSAR:

- PLATFORM_LEVEL: the Executable represents software on the platform level (i.e. conceptually located *on the level of* the middleware).

- APPLICATION_LEVEL: the Executable represents software on the application level (i.e. conceptually located *above* the middleware).

⌋()

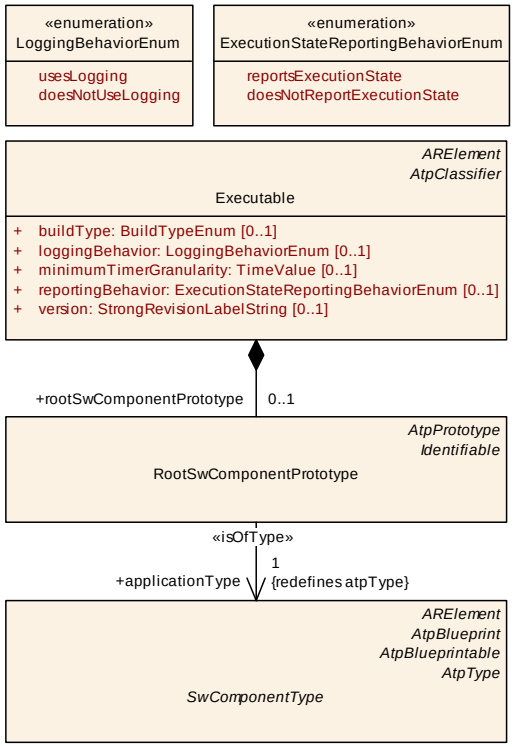


Figure 3.68: Modeling of the Executable

Class	**Executable**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This meta-class represents an executable program. **Tags:** atp.Status=draft atp.recommendedPackage=Executables			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
buildType	BuildTypeEnum	0..1	attr	This attribute describes the buildType of a module and/or platform implementation.
loggingBehavior	LoggingBehaviorEnum	0..1	attr	This attribute indicates the intended logging behavior of the enclosing Executable.
minimumTimer Granularity	TimeValue	0..1	attr	This attribute describes the minimum timer resolution (TimeValue of one tick) that is required by the Executable. **Tags:**atp.Status=draft
reporting Behavior	ExecutionState ReportingBehavior Enum	0..1	attr	this attribute controls the execution state reporting behavior of the enclosing Executable.

▽

△

Class	**Executable**			
rootSw Component Prototype	RootSwComponent Prototype	0..1	aggr	This represents the root SwCompositionPrototype of the Executable. This aggregation is required (in contrast to a direct reference of a SwComponentType) in order to support the definition of instanceRefs in Executable context. **Tags:**atp.Status=draft
version	StrongRevisionLabel String	0..1	attr	Version of the executable. **Tags:**atp.Status=draft

Table 3.127: Executable

Enumeration	**BuildTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation
Note	This enumeration defines the possible buildTypes a software module may be implemented. **Tags:**atp.Status=draft
Literal	*Description*
buildTypeDebug	Used for debugging. **Tags:**atp.EnumerationLiteralIndex=1
buildTypeRelease	Used for releasing. **Tags:**atp.EnumerationLiteralIndex=0

Table 3.128: BuildTypeEnum

Enumeration	**LoggingBehaviorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure
Note	This enumeration provides options for controlling of whether an Executable uses logging. **Tags:**atp.Status=draft
Literal	*Description*
doesNotUse Logging	The Executable indicates its intention to not use logging. **Tags:**atp.EnumerationLiteralIndex=0
usesLogging	The Executable indicates its intention to use logging **Tags:**atp.EnumerationLiteralIndex=1

Table 3.129: LoggingBehaviorEnum

Enumeration	**ExecutionStateReportingBehaviorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure
Note	This enumeration provides options for controlling of how an Executable reports its execution state to the Execution Management **Tags:**atp.Status=draft
Literal	*Description*
doesNotReport ExecutionState	The Executable shall not report its execution state to the Execution Management. **Tags:**atp.EnumerationLiteralIndex=1

▽

△

Enumeration	**ExecutionStateReportingBehaviorEnum**
reportsExecution State	The Executable shall report its execution state to the Execution Management. **Tags:**atp.EnumerationLiteralIndex=0

Table 3.130: ExecutionStateReportingBehaviorEnum

[TPS_MANI_01271]{DRAFT} **Semantics of `Executable.loggingBehavior`** ⌈Attribute `Executable.loggingBehavior` shall be used to indicate whether the enclosing `Executable` uses logging.

If the attribute does not exist, the `Executable` indicates that it does not use logging.⌋ *(RS_MANI_00023)*

[TPS_MANI_03056]{DRAFT} **Optionality of `Executable.rootSwComponentPrototype`** ⌈The aggregation `Executable.rootSwComponentPrototype` has been made optional in order to support the implementation of *platform modules* that do not utilize any service oriented communication and don't require any further formalization.⌋ *(RS_MANI_00023)*

Class	**RootSwComponentPrototype**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	The RootSwCompositionPrototype represents the top-level-composition of software components within an Executable. The contained SwComponentPrototypes are fully specified by their SwComponentTypes (including Port Prototypes, PortInterfaces, VariableDataPrototypes, etc.). **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
applicationType	SwComponentType	1	tref	This SwComponentType acts as the Type of the RootSw ComponentPrototype. **Stereotypes:** isOfType **Tags:**atp.Status=draft

Table 3.131: RootSwComponentPrototype

[constr_1492]{DRAFT} **`SwComponentType` referenced in the role `Executable.rootSwComponentPrototype.applicationType`** ⌈Any `SwComponentType` referenced in the role `Executable.rootSwComponentPrototype.applicationType`, or used to type a `SwComponentPrototype` nested inside the `SwComponentType` referenced in the role `Executable.rootSwComponentPrototype.applicationType` shall **only** be either a `CompositionSwComponentType` or an `AdaptiveApplicationSwComponentType`.⌋*()*

The example depicted in Figure 3.69 exemplifies the statement of [constr_1492]. The example shows a component hierarchy that consists of `SwComponentPrototype`s that are excursively typed by either a `CompositionSwComponentType` or an `AdaptiveApplicationSwComponentType`.

Class	**SwComponentType** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for AUTOSAR software components.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	AdaptiveApplicationSwComponentType, *AtomicSwComponentType*, CompositionSwComponentType, ParameterSwComponentType			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
port	PortPrototype	*	aggr	The PortPrototypes through which this SwComponent Type can communicate. The aggregation of PortPrototype is subject to variability with the purpose to support the conditional existence of PortPrototypes. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=port.shortName, port.variationPoint.short Label vh.latestBindingTime=preCompileTime
portGroup	PortGroup	*	aggr	A port group being part of this component. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
swComponent Documentation	SwComponent Documentation	0..1	aggr	This adds a documentation to the SwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=swComponentDocumentation, sw ComponentDocumentation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=-10

Table 3.132: SwComponentType

Class	**CompositionSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	A CompositionSwComponentType aggregates SwComponentPrototypes (that in turn are typed by Sw ComponentTypes) as well as SwConnectors for primarily connecting SwComponentPrototypes among each others and towards the surface of the CompositionSwComponentType. By this means hierarchical structures of software-components can be created. **Tags:**atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
component	SwComponent Prototype	*	aggr	The instantiated components that are part of this composition. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=component.shortName, component.variation Point.shortLabel vh.latestBindingTime=postBuild

△

Class	**CompositionSwComponentType**			
connector	SwConnector	*	aggr	SwConnectors have the principal ability to establish a connection among PortPrototypes. They can have many roles in the context of a CompositionSwComponentType. Details are refined by subclasses. The aggregation of SwConnectors is subject to variability with the purpose to support variant data flow. The aggregation is marked as atpSplitable in order to allow the extension of the ECU extract with AssemblySw Connectors between ApplicationSwComponentTypes and ServiceSwComponentTypes during the ECU integration. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=connector.shortName, connector.variation Point.shortLabel vh.latestBindingTime=postBuild
constantValue Mapping	ConstantSpecification MappingSet	*	ref	Reference to the ConstantSpecificationMapping to be applied for initValues of PPortComSpecs and RPortCom Spec. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=constantValueMapping
dataType Mapping	DataTypeMappingSet	*	ref	Reference to the DataTypeMapping to be applied for the used ApplicationDataTypes in ServiceInterfaces. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=dataTypeMapping

Table 3.133: CompositionSwComponentType

While the left part of Figure 3.69 resembles the modeling in the meta-model, the right part uses a simplified notation to give an idea how the nested definition of software-components could look like.

An obvious consequence of [constr_1492] is that no software-component that could be used on the *AUTOSAR classic platform* is allowed on the *AUTOSAR adaptive platform*, i.e. in the context of an `Executable.rootSwComponentPrototype.applicationType`.

Software-components on the *AUTOSAR adaptive platform* are mainly defined by their interaction with the outside world by means of `PortPrototype`s typed by `ServiceInterface`s. The definition of an internal behavior, with a minor exception, is not foreseen.

This lack of internal structure, in combination with decisions made regarding the scope of the generation of header files, leads to a situation where the implementation of a software component in source code is (in comparison to the situation on the *AUTOSAR classic platform*) way less subject to a strict separation.

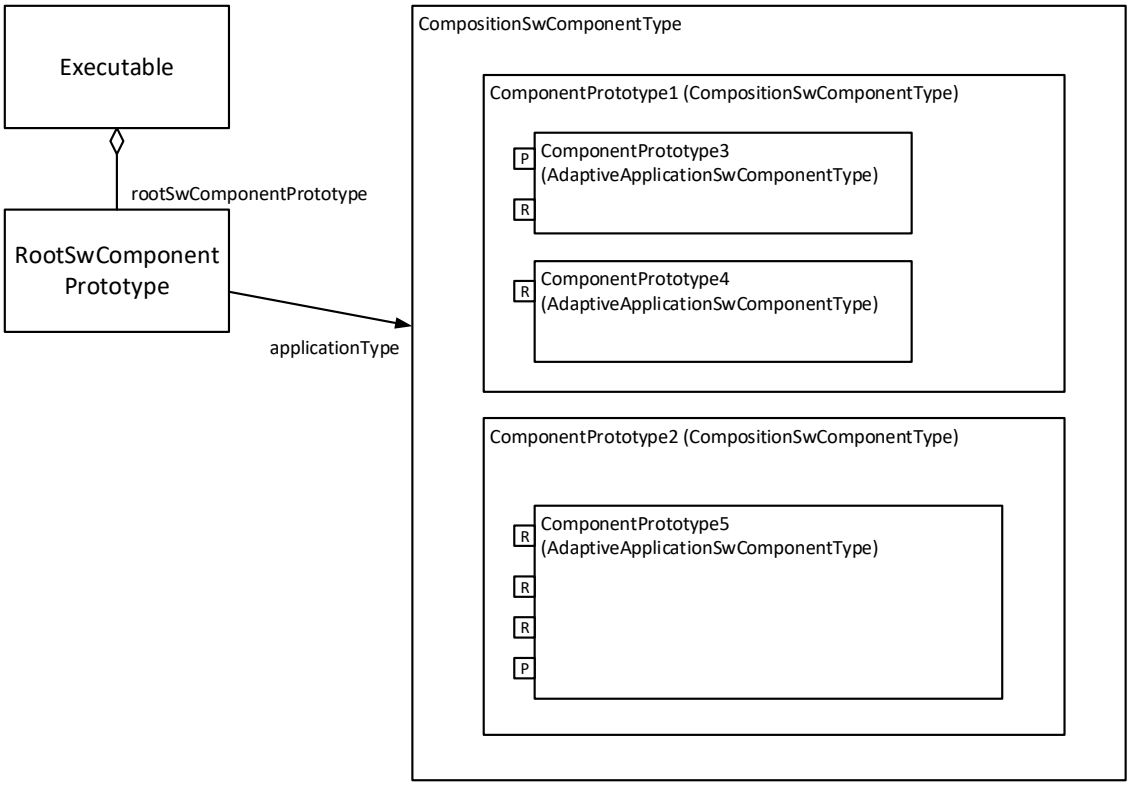


Figure 3.69: Example of the possible structure of an `Executable`

In other words, there is no real motivation to implement software-components separately from each other. It would be possible, although not encouraged, to implement all software-components of a given executable program directly within the `Main()` function of the program.

3.17 Optional Members in complex Data Structures

3.17.1 Background

The *AUTOSAR adaptive platform* supports the usage of a `TLV`[6] data encoding on the SOME/IP transport layer. `TLV` is typically used where at least a part of the transmitted data is only *optionally* existing and filled with meaningful values.

In other words: an optional part of a data structure may exist and carry meaningful values in one instance of data transmission and be completely missing in another instance of the data transmission.

The receiving software needs to be able to identify whether the optional part exists and read its value accordingly.

[6]This abbreviation stands for tag-length-value

The receiving software also needs to be able to still execute meaningfully if the optional part of such a data structure does not exist in the specific communication instance.

Consequently, it is necessary to be able to precisely identify the parts of a data structure that may become optional for specific instances of data transmission.

In terms of the AUTOSAR meta-model, the identification could - in principle - be attached at various levels of abstraction:

AutosarDataType In this case the optionality that is primarily only needed for communication purposes would still be existing in all other usages of data types. AUTOSAR still sees use cases for implementing this option, especially in the context of the *AUTOSAR classic platform*.

Admittedly, the definition of different optionality configurations for the same data type may lead to the existence of a bunch of structurally identical data types that only vary in terms of optionality. The existence of variation points may help to mitigate this effect, though.

ServiceInterface In this case the optionality is defined where it is actually required. However, different optionality could - in principle - be defined for DataPrototypes typed by the same AutosarDataType.

This would lead to an increased effort for the definition of C++ data types in the context of the same ServiceInterface. Additional constraints have been identified in the context of the *AUTOSAR classic platform* that finally render this option as not viable.

ComSpec In this case the definition of optionality would even be more specific in comparison to the definition of optionality on the level of ServiceInterfaces.

On top of that, the task to define optionality in the vast majority of cases is done by an OEM, whereas the model definition on the level of ComSpec requires the existence of SwComponentTypes and this definition is in many cases in the domain of a supplier.

As a result of this consideration, AUTOSAR has opted for implementation the concept of defining the optionality on the level of the AutosarDataType.

3.17.2 Definition of Optionality

As mentioned before, the concrete definition of optionality on the level of an AutosarDataType is done by the indication of individual elements of the composite AutosarDataType.

More specifically, the definition of optionality needs to be supported for subclasses of AutosarDataType, namely on the level of ApplicationDataType as well as on the level of CppImplementationDataType.

In other words, if ApplicationDataTypes with optional elements are used to define a ServiceInterface then it is still necessary to convey the optionality down to the level of data type definition that directly affects the language binding of the AUTOSAR model.

Figure 3.70 shows the modeling of optionality on the level of ApplicationDataType.

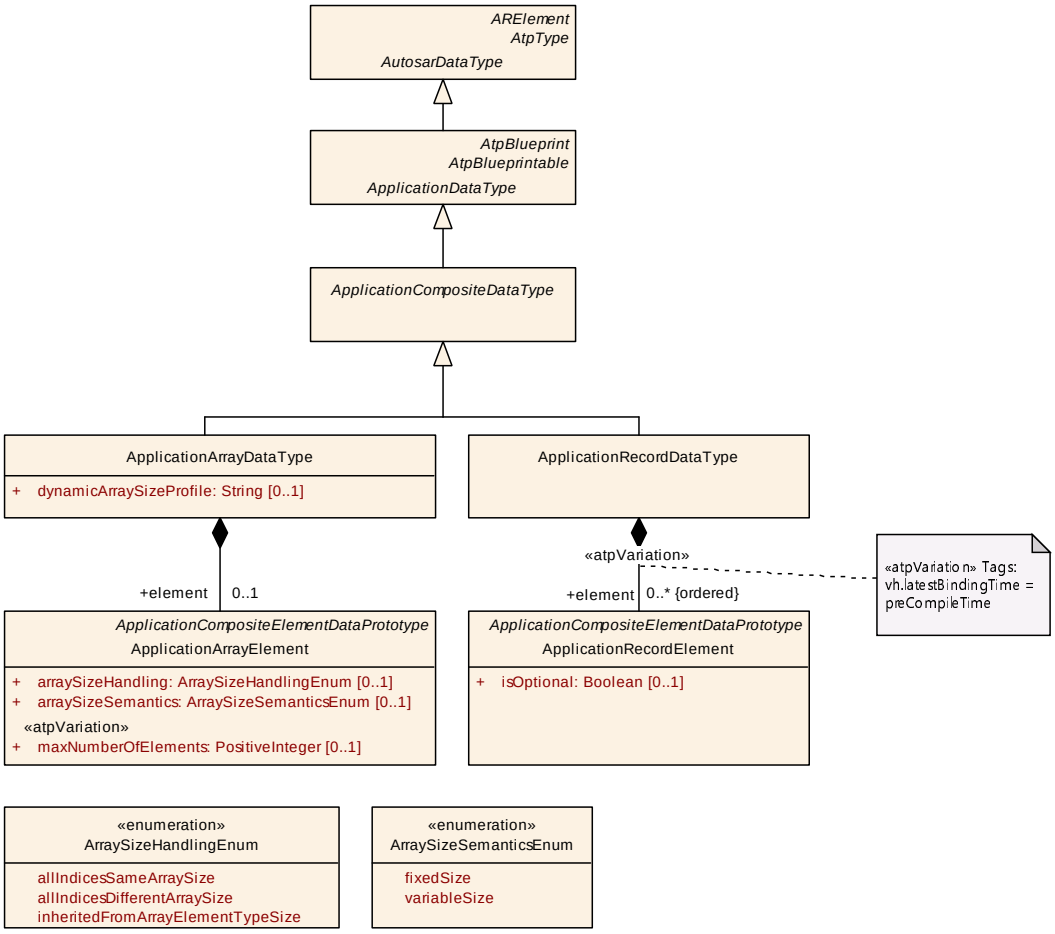


Figure 3.70: Modeling of optionality on the level of ApplicationDataType

[TPS_MANI_01184]{DRAFT} **Definition of optional elements on the level of ApplicationDataType** ⌈The modeling approach for the definition of optional elements on the level of ApplicationDataType is to set the attribute ApplicationRecordElement.isOptional to the value True.

If the attribute is not set or set to the value False then the respective ApplicationRecordElement **shall be considered mandatory**.⌋*(RS_MANI_00030)*

Class	ApplicationRecordDataType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	An application data type which can be decomposed into prototypes of other application data types. **Tags:**atp.recommendedPackage=ApplicationDataTypes			
Base	*ARElement*, *ARObject*, *ApplicationCompositeDataType*, *ApplicationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
element (ordered)	ApplicationRecordElement	*	aggr	Specifies an element of a record. The aggregation of ApplicationRecordElement is subject to variability with the purpose to support the conditional existence of elements inside a ApplicationrecordDataType. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime

Table 3.134: ApplicationRecordDataType

Class	ApplicationRecordElement			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Describes the properties of one particular element of an application record data type.			
Base	*ARObject*, *ApplicationCompositeElementDataPrototype*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing ApplicationRecordElement as optional. This means the that, at runtime, the ApplicationRecord Element may or may not have a valid value and shall therefore be ignored. The underlying runtime software provides means to set the ApplicationRecordElement as not valid at the sending end of a communication and determine its validity at the receiving end.

Table 3.135: ApplicationRecordElement

On top of that, it is still possible to use `CppImplementationDataType` directly for the definition of a `ServiceInterface`.

[TPS_MANI_01185]{DRAFT} **Definition of optional elements on the level of `CppImplementationDataType`** ⌈The modeling approach for the definition of optional elements on the level of `CppImplementationDataType` is to set the attribute `CppImplementationDataTypeElement.isOptional` to the value `True`.

If the attribute is not set or set to the value `False` then the respective `CppImplementationDataTypeElement` **shall be considered mandatory**.⌋*(RS_MANI_00030)*

The attribute `NotAvailableValueSpecification.defaultPattern` has no meaning for the initialization of `DataPrototype`s on the *AUTOSAR adaptive platform*. This aspect is covered by [TPS_MANI_01333]:

[TPS_MANI_01333]{DRAFT} **Attribute `NotAvailableValueSpecification.defaultPattern` is not applicable** ⌈The attribute `NotAvailableValueSpecification.defaultPattern` (if defined) shall be ignored by the adaptive platform.

The rationale for ignoring the `defaultPattern` is that the optional data is technically not accessible from application code in case it has not been received.⌋*(RS_MANI_00030)*

3.18 Serialization Properties

In Adaptive AUTOSAR, the serialization code is generated out of the service description and is compiled and executed in the application context.

The meta-class `TransformationPropsToServiceInterfaceElementMapping` defines the serialization for a `ServiceInterface` element and provides the necessary serialization settings with the `TransformationProps` element.

The existence of a `TransformationPropsToServiceInterfaceElementMapping` demands the existence of serialization code that is linked with the application component object file to an application binary.

The serialization of SOME/IP is based on the `ServiceInterface` specification. If an `AutosarDataPrototype` that is used within a `ServiceInterface` is composite like a structure, union or array then SOME/IP supports the configuration of length fields that will be put in front of the serialized data.

AUTOSAR supports the configuration of such serialization settings on two different levels:

- Modeling on `ServiceInterface` element level that is valid for all available occurrences of a `DataPrototype` in the `ServiceInterface` element. This case is described in detail in chapter 3.18.1.
- Fine granular modeling on the level of `DataPrototype`s described in this chapter. This case is described in detail in chapter 3.18.2.

3.18.1 Default Values for Serialization Properties

[TPS_MANI_03101]{DRAFT} **SOME/IP serialization** ⌈The `ApSomeipTransformationProps` meta-class that is referenced by the `TransformationPropsToServiceInterfaceElementMapping` in the role `transformationProps` provides the ability to define a SOME/IP serialization settings for `ServiceInterface` elements that are referenced by the `TransformationPropsToServiceInterfaceElementMapping` in the role `event`, `method` or `field`.⌋*(RS_MANI_00008, RS_MANI_00025)*

[constr_3395]{DRAFT} **TransformationPropsToServiceInterfaceElementMapping is restricted to one single ServiceInterface** ⌈All ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field shall be aggregated by the same ServiceInterface in the role event, method or field.⌋ *()*

[TPS_MANI_03103]{DRAFT} **Default size for all array and map length fields** ⌈The attribute sizeOfArrayLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available variable size arrays (vectors), fixed size arrays and associative_maps defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.⌋ *(RS_MANI_00008, RS_MANI_00025)*

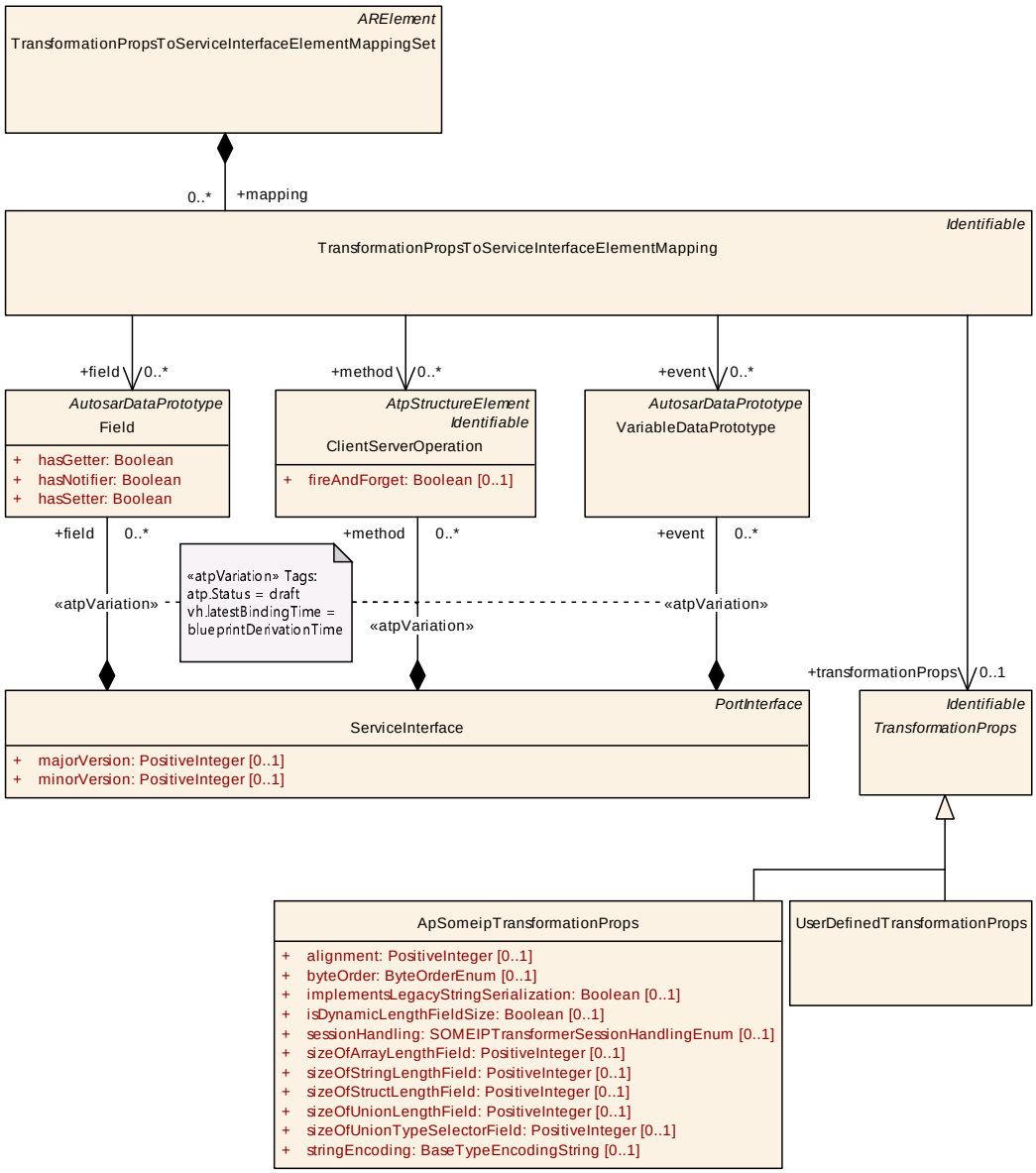


Figure 3.71: Association of serialization properties with a ServiceInterface

[TPS_MANI_03104]{DRAFT} **Default size for all structure length fields** ⌈The attribute sizeOfStructLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available structures defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03117]{DRAFT} **Default size for all string length fields** ⌈The attribute sizeOfStringLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available strings defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03105]{DRAFT} **Default size for all union length fields** ⌈The attribute sizeOfUnionLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available unions defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03106]{DRAFT} **Default size for all union type selector fields** ⌈The attribute sizeOfUnionTypeSelectorField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a type field generated by SOME/IP in front of all available unions defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03107]{DRAFT} **Default alignment for all dynamic DataPrototypes** ⌈The attribute alignment of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the padding for alignment purposes that will be added by SOME/IP after the serialized data of all variable data length data elements defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03108]{DRAFT} **Default Byte Order for all DataPrototypes** ⌈The attribute byteOrder of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the Byte Order in the serialized data stream resulting from

ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.⌋ *(RS_MANI_00008, RS_MANI_00025)*

[constr_1614]{DRAFT} **Existence of attribute TransformationPropsToServiceInterfaceElementMapping.transformationProps.sessionHandling** ⌈The attribute ApSomeipTransformationProps.sessionHandling shall only exist if the TransformationPropsToServiceInterfaceElementMapping that refers to the respective ApSomeipTransformationProps in the role transformationProps does **not** refer to a ClientServerOperation in the role method.⌋ *()*

[TPS_MANI_01210]{DRAFT} **Default encoding for all DataPrototypes typed by CppImplementationDataType of category STRING** ⌈The attribute stringEncoding of a ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the string encoding in the serialized data stream resulting from ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.⌋ *(RS_MANI_00008, RS_MANI_00025)*

[constr_1675]{DRAFT} **Existence of attribute ApSomeipTransformationProps.stringEncoding** ⌈The attribute TransformationPropsToServiceInterfaceElementMapping.transformationProps.stringEncoding shall only exist for a event, method or field (referenced by the same TransformationPropsToServiceInterfaceElementMapping) that consists of or contains a DataPrototype typed by a CppImplementationDataType of category STRING.⌋ *()*

Please note that more details about ApSomeipTransformationProps can be found in chapter 3.18.2.

[constr_1678]{DRAFT} **Allowed values for attribute ApSomeipTransformationProps.stringEncoding** ⌈Imposed by technical restrictions in the definition of the SOME/IP message format [7], only two possible values of attribute ApSomeipTransformationProps.stringEncoding are allowed:

- UTF-8: UCS Transformation Format 8
- UTF-16: Character encoding for Unicode *code points* based on 16 bit *code units* [16]

⌋ *()*

Class	**ApSomeipTransformationProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::SerializationProperties			
Note	SOME/IP serialization properties. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TransformationProps*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
alignment	PositiveInteger	0..1	attr	Defines the padding for alignment purposes that will be added by the SOME/IP transformer after the serialized data of the variable data length data element. The alignment shall be specified in Bits.
byteOrder	ByteOrderEnum	0..1	attr	Specifies the byte order of data in the serialized data stream.
implements LegacyString Serialization	Boolean	0..1	attr	This attribute indicates that Strings in the SOME/IP message shall NOT be serialized according to the SOME/IP specification for Strings. If this attribute is set to true, BOM and null-termination shall NOT be added in the serialization for Strings in the payload. If this attribute is set to false (or not set) BOM and null-termination shall be added in the serialization for Strings in the payload according to the SOME/IP specification for Strings. NOTE! This attribute is not future safe, and will be removed in an upcoming AUTOSAR release!
isDynamic LengthFieldSize	Boolean	0..1	attr	This attribute represents the ability to control the setting of the wire type for TLV encoding. If the attribute is set to True then wire type 5-7 shall be used. If the attribute does not exist or is set to False then wire type 4 shall be used.
session Handling	SOMEIPTransformer SessionHandlingEnum	0..1	attr	Defines whether the SOME/IP transformer shall use session handling for Sender/Receiver communication.
sizeOfArray LengthField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of a variable size Array (Vector), fixed-size Array or an Associative_Map. It describes the size of the length field (in Bytes) that will be put in front of the Array or Associative_Map in the SOME/IP message.
sizeOfString LengthField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of a String. It describes the size of the length field (in Bytes) that will be put in front of the String in the SOME/IP message.
sizeOfStruct LengthField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of an Struct. It describes the size of the length field (in Bytes) that will be put in front of the Struct in the SOME/IP message.
sizeOfUnion LengthField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of a Union. It describes the size of the length field (in Bytes) that will be put in front of the Union in the SOME/IP message.
sizeOfUnion TypeSelector Field	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of a Union. It describes the size of the type selector field (in Bytes) that will be put in front of the Union in the SOME/IP message.

△

Class	ApSomeipTransformationProps			
stringEncoding	BaseTypeEncoding String	0..1	attr	Configures the encoding for SOME/IP serialization for the referenced dataPrototype in case of an String.

Table 3.136: ApSomeipTransformationProps

[TPS_MANI_03102]{DRAFT} **UserDefined serialization** ⌈The `UserDefinedTransformationProps` meta-class that is referenced by the `TransformationPropsToServiceInterfaceElementMapping` in the role `transformationProps` provides the ability to define a User defined serialization for `ServiceInterface` elements that are referenced by the `TransformationPropsToServiceInterfaceElementMapping` in the role `event`, `method` or `field`.⌋*(RS_MANI_00014, RS_MANI_00025)*

Please note that `UserDefinedTransformationProps` is derived from meta-class `Identifiable` and therefore has the ability to describe special data (`sdg`) by which it is possible to define custom structural extensions of an AUTOSAR model in a generic way. For more information about special data please refer to [6].

Class	TransformationPropsToServiceInterfaceElementMappingSet			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::SerializationProperties			
Note	Collection of TransformationPropsToServiceInterfaceElementMappings. **Tags:** atp.Status=draft atp.recommendedPackage=TransformationPropsToServiceInterfaceMappingSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
mapping	TransformationPropsTo ServiceInterface ElementMapping	*	aggr	Mapping that assigns serialization properties to elements of a ServiceInterface. **Tags:**atp.Status=draft

Table 3.137: TransformationPropsToServiceInterfaceElementMappingSet

Class	TransformationPropsToServiceInterfaceElementMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This meta-class represents the ability to associate a ServiceInterface element with TransformationProps. The referenced elements of the Service Interface will be serialized according to the settings defined in the TransformationProps. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
event	VariableDataPrototype	*	ref	This represents the reference to one or several events of one ServiceInterface. **Tags:**atp.Status=draft

▽

△

Class	TransformationPropsToServiceInterfaceElementMapping			
field	Field	*	ref	This represents the reference to one or several fields of one ServiceInterface. **Tags:**atp.Status=draft
method	ClientServerOperation	*	ref	This represents the reference to one or several methods of one ServiceInterface. **Tags:**atp.Status=draft
tlvDataId Definition	TlvDataIdDefinitionSet	*	ref	This reference identifies the TlvDataIdDefinitions relevant for the enclosing TransformationPropsToServiceInterface Mapping. **Tags:**atp.Status=draft
transformation Props	TransformationProps	0..1	ref	This represents the reference to the applicable Serialization properties. **Tags:**atp.Status=draft

Table 3.138: TransformationPropsToServiceInterfaceElementMapping

Class	UserDefinedTransformationProps			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	The class UserDefinedTransformationProps specifies specific configuration properties of a user defined serializer.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TransformationProps*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.139: UserDefinedTransformationProps

3.18.2 Individual Definition of Serialization Properties

[TPS_MANI_03109]{DRAFT} **`TransformationProps` on the level of `DataPrototype`s overwrites `TransformationProps` settings on the level of a `ServiceInterface`** ⌈The fine granular modeling of `TransformationProps` on the level of `DataPrototype`s overwrites the `TransformationProps` settings defined on the level of a `ServiceInterface` described with the `TransformationPropsToServiceInterfaceElementMappingSet`.⌋*()*

[constr_3361]{DRAFT} **Selective definition of serialization settings** ⌈If a `SomeipDataPrototypeTransformationProps` is defined for a composite `DataPrototype` of an element of a `ServiceInterface` (`method`, `field`, `event`) and if the reference `someipTransformationProps` exists then `SomeipDataPrototypeTransformationProps` that define the reference `someipTransformationProps` shall be defined for all other composite `DataPrototype`s of the `ServiceInterface` element as well.⌋*()*

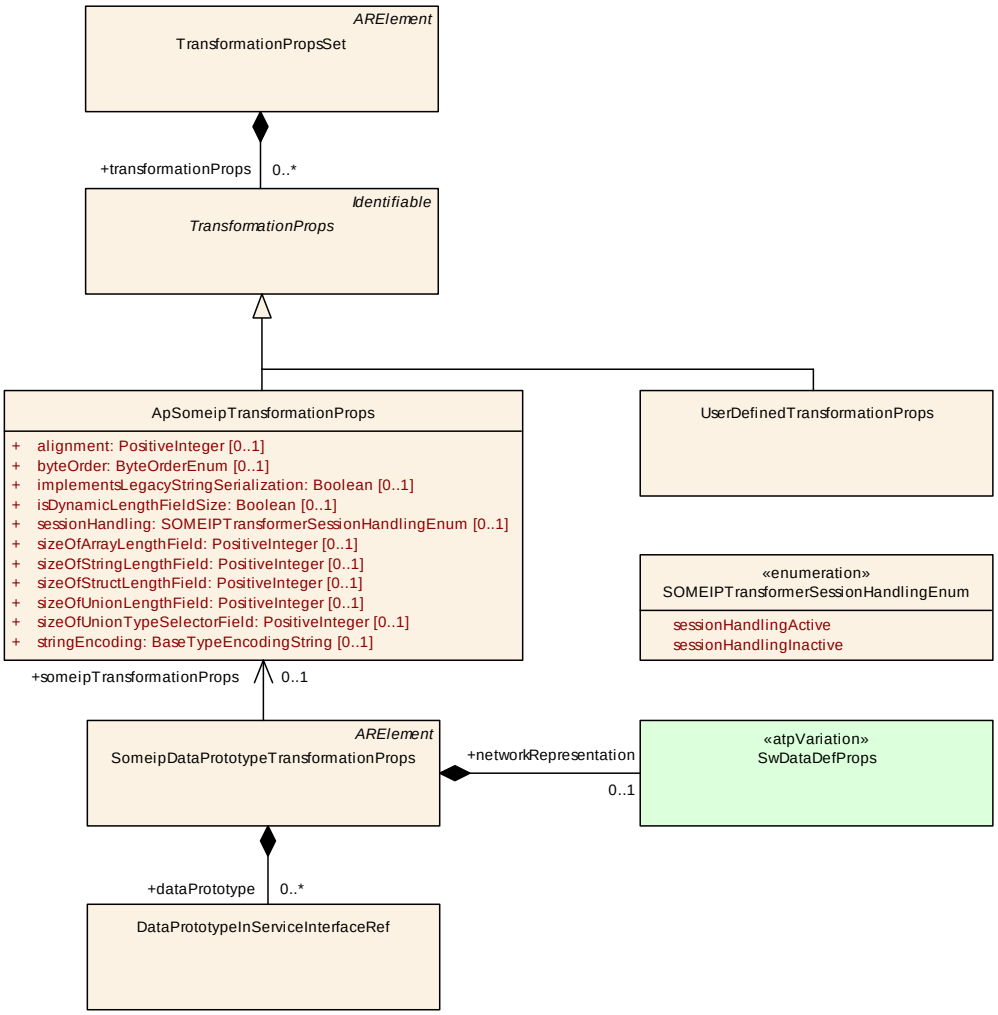


Figure 3.72: Overview about SOME/IP Serialization Properties

Class	**TransformationPropsSet**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	Collection of TransformationProps. **Tags:**atp.recommendedPackage=TransformationPropsSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
transformation Props	TransformationProps	*	aggr	Transformer specific configuration properties.

Table 3.140: TransformationPropsSet

Enumeration	**SOMEIPTransformerSessionHandlingEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer
Note	Enables or disable session handling for SOME/IP transformer
Literal	***Description***
sessionHandling Active	The SOME/IP Transformer shall use session handling **Tags:**atp.EnumerationLiteralIndex=0
sessionHandling Inactive	The SOME/IP Transformer doesn't use session handling **Tags:**atp.EnumerationLiteralIndex=1

Table 3.141: SOMEIPTransformerSessionHandlingEnum

[TPS_MANI_03070]{DRAFT} **Size of a length field for a chosen array or map** ⌈The attribute `sizeOfArrayLengthField` of `ApSomeipTransformationProps` defines the size of a length field generated by SOME/IP in front of a variable size array (vector), fixed size array or associative_map for which the `SomeipDataPrototypeTransformationProps` is defined, i.e. the variable size array (vector), fixed size array or associative_map that is referenced within the aggregated `DataPrototypeInServiceInterfaceRef`.⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3353]{DRAFT} **Restriction in usage of `ApSomeipTransformationProps.sizeOfArrayLengthField`** ⌈The value of the attribute `sizeOfArrayLengthField` shall be either 0, 1, 2 or 4.⌋*()*

[constr_3447]{DRAFT} **`ApSomeipTransformationProps.sizeOfArrayLengthField` that equals 0** ⌈The `sizeOfArrayLengthField` value of 0 is only allowed to be used if a fixed size array for which the `SomeipDataPrototypeTransformationProps` is defined is referenced within the aggregated `DataPrototypeInServiceInterfaceRef`.⌋*()*

The setting of `sizeOfArrayLengthField` for fixed size arrays supports a backward compatible extension of such arrays with additional array elements.

[TPS_MANI_03071]{DRAFT} **Size of a length field for a chosen structure** ⌈The attribute `sizeOfStructLengthField` of `ApSomeipTransformationProps` defines the size of a length field generated by SOME/IP in front of a structure for which the `SomeipDataPrototypeTransformationProps` is defined, i.e. the structure that is referenced within the aggregated `DataPrototypeInServiceInterfaceRef`.⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3354]{DRAFT} **Restriction in usage of `ApSomeipTransformationProps.sizeOfStructLengthField`** ⌈The value of the attribute `sizeOfStructLengthField` shall be either 0, 1, 2 or 4.⌋*()*

[TPS_MANI_03116]{DRAFT} **Size of a length field for a chosen string** ⌈The attribute `sizeOfStringLengthField` of `ApSomeipTransformationProps` defines the size of a length field generated by SOME/IP in front of a String for which the `SomeipDataPrototypeTransformationProps` is defined, i.e. the String that is referenced within the aggregated `DataPrototypeInServiceInterfaceRef`.⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3372]{DRAFT} **Restriction in usage of ApSomeipTransformationProps.sizeOfStringLengthField** ⌈The value of the attribute sizeOfStringLengthField shall be either 0, 1, 2 or 4.⌋*()*

[TPS_MANI_03217]{DRAFT} **On-the-wire encoding for a chosen string** ⌈The attribute stringEncoding of ApSomeipTransformationProps defines the on-the-wire encoding of a String for which the SomeipDataPrototypeTransformationProps is defined, i.e. the String that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.⌋*(RS_MANI_00008, RS_MANI_00024)*

[TPS_MANI_03072]{DRAFT} **Size of a length field for a chosen union** ⌈The attribute sizeOfUnionLengthField of ApSomeipTransformationProps defines the size of a length field generated by SOME/IP in front of a union for which the SomeipDataPrototypeTransformationProps is defined, i.e. the union that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3355]{DRAFT} **Restriction in usage of ApSomeipTransformationProps.sizeOfUnionLengthField** ⌈The value of the attribute sizeOfUnionLengthField shall be either 0, 1, 2 or 4.⌋*()*

[TPS_MANI_03073]{DRAFT} **Alignment of a dynamic DataPrototype** ⌈The attribute alignment of ApSomeipTransformationProps defines the padding for alignment purposes that will be added by SOME/IP after the serialized data of the variable data length data element for which the SomeipDataPrototypeTransformationProps is defined, i.e. the variable data length DataPrototype that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3356]{DRAFT} **Restriction in usage of ApSomeipTransformationProps.alignment** ⌈The value of the attribute alignment shall be either 8, 16, 32, 64, 128, or 256.⌋*()*

[TPS_MANI_03074]{DRAFT} **Size of a type selector field for a chosen union** ⌈The attribute sizeOfUnionTypeSelectorField of ApSomeipTransformationProps defines the size of a type selector field generated by SOME/IP in front of a union for which the SomeipDataPrototypeTransformationProps is defined, i.e. the union that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3357]{DRAFT} **Restriction in usage of ApSomeipTransformationProps.sizeOfUnionTypeSelectorField** ⌈The value of the attribute sizeOfUnionTypeSelectorField shall be either 1, 2 or 4.⌋*()*

[TPS_MANI_03075]{DRAFT} **Byte Order of chosen DataPrototype in the serialized data stream** ⌈The attribute byteOrder of ApSomeipTransformationProps defines the Byte Order in front of the DataPrototype in the serialized data stream for which the SomeipDataPrototypeTransformationProps is defined, i.e. the DataPrototype that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.⌋*(RS_MANI_00008, RS_MANI_00024)*

[TPS_MANI_03235]{DRAFT} **Usage of `ApSomeipTransformationProps.sessionHandling`** ⌈The `sessionHandling` attribute defined in an `ApSomeipTransformationProps` that is referenced by `SomeipDataPrototypeTransformationProps` is not relevant for the `DataPrototype`s that are referenced in the `SomeipDataPrototypeTransformationProps`.⌋*(RS_MANI_00008, RS_MANI_00024)*

The `sessionHandling` attribute is used for the activation/deactivation of the SessionHandling for Events/Notifiers and therefore the usage via `TransformationPropsToServiceInterfaceElementMapping` is the only valid configuration option.

Class	**SomeipDataPrototypeTransformationProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::SerializationProperties			
Note	This meta-class represents the ability to define data transformation props specifically for a SOME/IP serialization for a given DataPrototype. **Tags:** atp.Status=draft atp.recommendedPackage=SomeipDataPrototypeTransformationPropss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataPrototype	DataPrototypeInService InterfaceRef	*	aggr	Collection of DataPrototypes for which the settings in SomeipDataPrototypeTransformationProps are valid. For reuse reasons the SomeipDataPrototypeTransformation Props is able to aggregate several DataPrototypes. **Tags:**atp.Status=draft
network Representation	SwDataDefProps	0..1	aggr	Optional specification of the actual network representation for the referenced primitive DataPrototype. If a network representation is provided then the baseType available in the SwDataDefProps shall be used as input for the serialization/deserialization. If the network Representation is not provided then the baseType of the AbstractImplementationDataType shall be used for the serialization/deserialization. **Tags:**atp.Status=draft
someip Transformation Props	ApSomeip TransformationProps	0..1	ref	This reference represents the ability to define data transformation props specifically for a SOME/IP serialization. **Tags:**atp.Status=draft

Table 3.142: SomeipDataPrototypeTransformationProps

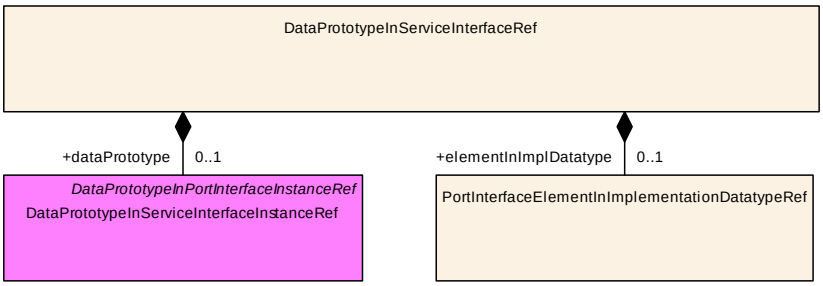


Figure 3.73: Reference to a DataPrototype in the context of a PortInterface that is typed by an ApplicationDataType or by a CppImplementationDataType

[TPS_MANI_01136]{DRAFT} **AutosarDataPrototype is the target of the DataPrototypeInServiceInterfaceRef** ⌈If the target of an DataPrototypeInServiceInterfaceRef is an AutosarDataPrototype the role DataPrototypeInServiceInterfaceRef.dataPrototype shall be used to describe the reference **independently** of whether the AutosarDataPrototype is typed by an ApplicationDataType or a CppImplementationDataType and even **independently** of whether the AutosarDataType of the AutosarDataPrototype represents a composite data type.⌋*(RS_MANI_00008, RS_MANI_00024)*

[TPS_MANI_01137]{DRAFT} **Applicable use cases for DataPrototypeInServiceInterfaceRef** ⌈Table 3.143 contains a comprehensive list of use cases for the usage of DataPrototypeInServiceInterfaceRef.⌋*(RS_MANI_00008, RS_MANI_00024)*

Use case	Role
AutosarDataPrototype typed by an ApplicationDataType	dataPrototype
DataPrototype in AutosarDataPrototype typed by an ApplicationCompositeDataType	dataPrototype
AutosarDataPrototype typed by a CppImplementationDataType	dataPrototype
DataPrototype in AutosarDataPrototype typed by a CppImplementationDataType	elementInImplDatatype

Table 3.143: Possible use cases for the usage of DataPrototypeInServiceInterfaceRef

From a careful observation of Table 3.143 it should be clear that there is no valid use case to simultaneously use the two roles dataPrototype and elementInImplDatatype in the context of the same DataPrototypeInServiceInterfaceRef.

[constr_1551]{DRAFT} **Existence of DataPrototypeInServiceInterfaceRef.dataPrototype vs. DataPrototypeInServiceInterfaceRef.elementInImplDatatype** ⌈For every given DataPrototypeInServiceInterfaceRef, either the aggregation DataPrototypeInServiceInterfaceRef.dataPrototype or DataPrototypeInServiceInterfaceRef.elementInImplDatatype shall exist.⌋*()*

The usage of the SomeipDataPrototypeTransformationProps.networkRepresentation is explained in more detail in the System Template [17] in [TPS_SYST_02136] and [TPS_SYST_02137].

3.18.3 Assignment of TLV properties

3.18.3.1 Assignment of TLV Data IDs

[TPS_MANI_01097]{DRAFT} **Assignment of TLV data ids** ⌈The assignment of TLV data ids is done in the context of the specification of `TransformationPropsToServiceInterfaceElementMapping`, namely by means of the attribute `TransformationPropsToServiceInterfaceElementMapping.tlvDataIdDefinition.id`.⌋*(RS_MANI_00030)*

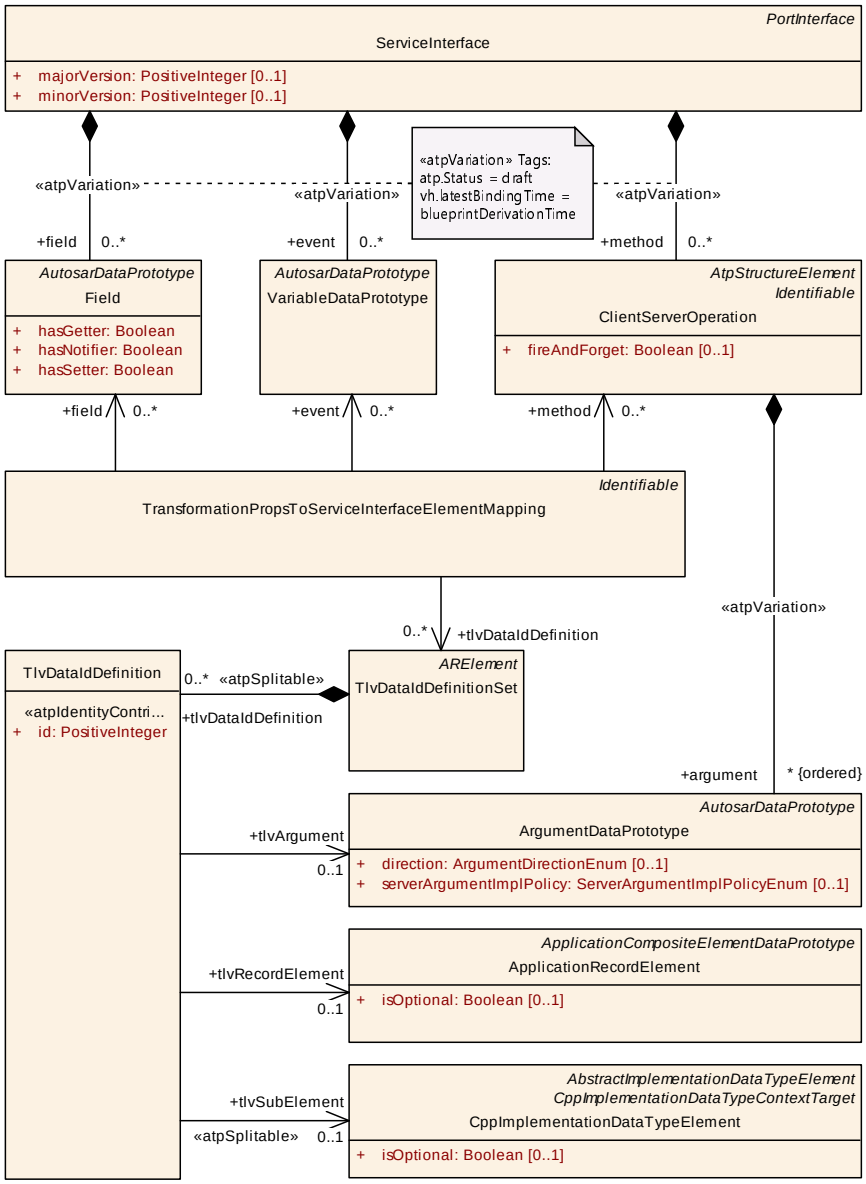


Figure 3.74: Modeling of the TLV data id

This approach takes benefit from the fact that the `TlvDataIdDefinition` is able to create references to relevant model elements.

The assignment of the TLV data id is therefore done by creating such a reference and assigning a TLV data id to it by means of the attribute `TlvDataIdDefinition.id`.

Please note that the assignment of TLV data ids is compulsory for an entire data structure that has at least one optional member. In a nutshell, this conclusion (that is also backed by [PRS_SOMEIP_00230], see [7]) is the motivation for the existence of [constr_1594], and [constr_1595].

Please note further that the assignment of TLV data ids is not restricted to data structures with optional members. There is also a use case to support sending the elements of a specific data structure in arbitrary order even if none of the elements is considered optional.

[TPS_MANI_01270]{DRAFT} **Reference from `TransformationPropsToServiceInterfaceElementMapping` to `TlvDataIdDefinitionSet`** ⌈The reference from `TransformationPropsToServiceInterfaceElementMapping` to `TlvDataIdDefinitionSet` means that it is in the hand of the creator of a model to decide whether a global scope should be assumed or whether the definition needs to be customized for a specific case.⌋*(RS_MANI_00030)*

Class	**TlvDataIdDefinitionSet**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	This meta-class acts as a container of TlvDataIdDefinitions to be used in a given context **Tags:**atp.recommendedPackage=TlvDataDefinitionSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tlvDataId Definition	TlvDataIdDefinition	*	aggr	This aggregation represents the collection of TlVDataTid Definitions aggregated by the TlvDataIdDefinitionSet **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=tlvDataIdDefinition.id

Table 3.144: TlvDataIdDefinitionSet

[constr_1594]{DRAFT} **Consistent assignment of TLV data ids to `ApplicationRecordDataType`** ⌈For every `ApplicationRecordDataType` where direct members set the attribute `ApplicationRecordElement.isOptional` to the value `True` references to **all direct members** of this `ApplicationRecordDataType` shall be created on the basis of the definition of `TlvDataIdDefinition`.⌋*()*

[constr_1595]{DRAFT} **Consistent assignment of TLV data ids to `CppImplementationDataType` or `CppImplementationDataTypeElement`** ⌈For every `CppImplementationDataType` of `category STRUCTURE` where direct members set the attribute `CppImplementationDataTypeElement.isOptional` to the value `True` references to **all direct members** of this `CppImplementationDataType` shall be created on the basis of the definition of `TlvDataIdDefinition`.⌋*()*

The definition of a `TlvDataIdDefinition` that refers to an eligible model element is not limited to scenarios where optional elements are defined. It is also possible to define `TlvDataIdDefinition` for arbitrary methods or data structures.

A typical use case could be to prepare the argument list or sub-elements for future extensions. However, if one argument or sub-element is referenced then it is necessary to define references from TlvDataIdDefinitions to all other arguments or sub-elements as well.

[constr_1593]{DRAFT} **Completeness of the existence of a set of TlvDataIdDefinition.tlvArguments** ⌈If the reference TlvDataIdDefinition.tlvArgument exists for one argument of a given ClientServerOperation then further TlvDataIdDefinition.tlvArgument shall exist **for all** arguments of the given ClientServerOperation and all affected TlvDataIdDefinition shall be aggregated by the same TransformationPropsToServiceInterfaceElementMapping.⌋*()*

[constr_1603]{DRAFT} **Completeness of the existence of a set of TlvDataIdDefinition.tlvRecordElements** ⌈If the reference TlvDataIdDefinition.tlvRecordElement exists for one element of a given ApplicationRecordDataType then further TlvDataIdDefinition.tlvRecordElement shall exist **for all** elements of the given ApplicationRecordDataType and all affected TlvDataIdDefinition shall be aggregated by the same TransformationPropsToServiceInterfaceElementMapping.⌋*()*

[constr_1604]{DRAFT} **Completeness of the existence of a set of TlvDataIdDefinition.tlvSubElements** ⌈If the reference TlvDataIdDefinition.tlvSubElement exists for one subElement of a given CppImplementationDataType or CppImplementationDataTypeElement then further TlvDataIdDefinition.tlvSubElement shall exist **for all** subElements of the given CppImplementationDataType or CppImplementationDataTypeElement and all affected TlvDataIdDefinition shall be aggregated by the same TransformationPropsToServiceInterfaceElementMapping.⌋*()*

Class	**TlvDataIdDefinition**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	This meta-class represents the ability to define the tlvDataId.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
id	PositiveInteger	1	attr	This attribute represents the definition of the value of the TlvDataId **Stereotypes:** atpIdentityContributor
tlvArgument	ArgumentDataPrototype	0..1	ref	This reference assigns a tlvDataId to a given argument of a ClientServerOperation.
tlvRecord Element	ApplicationRecord Element	0..1	ref	This reference associates the definition of a TLV data id with a given ApplicationRecordElement.

▽

△

Class	TlvDataIdDefinition			
tlvSubElement	CppImplementation DataTypeElement	0..1	ref	This reference associates the definition of a TLV data id with a given CppImplementationDataTypeElement. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=tlvSubElement atp.Status=draft

Table 3.145: TlvDataIdDefinition

The definition of a `TlvDataIdDefinition.id` has the purpose to provide means to unambiguously identify the argument or sub-element. For this purpose, the value of the `id` needs to be unique in the respective context.

[constr_1596]{DRAFT} **Scope of the uniqueness of the value of `TlvDataIdDefinition.id` for references to `ArgumentDataPrototype`** ⌈For all `TlvDataIdDefinition` that are referencing `ArgumentDataPrototype`s of a given `ClientServerOperation` in the role `tlvArgument`, the attribute `TlvDataIdDefinition.id` **shall exist and have a unique value per communication direction**, i.e. in the context of the collection of all

- `argument`s where attribute `direction` is set to either `in` or `inout`
- `argument`s where attribute `direction` is set to either `out` or `inout`
- `argument`s where attribute `direction` is set to `inout` (if the `method` **only** has `argument`s where attribute `direction` is set to `inout`)

of the respective enclosing `ClientServerOperation`.⌋*()*

Rationale for the existence of [constr_1596]: `argument`s where attribute `direction` is set to either `in` or `inout` are never sent in the same SOME/IP message as `argument`s where attribute `direction` is set to either `out` or `inout`.

[constr_1597]{DRAFT} **Scope of the uniqueness of the value of `TlvDataIdDefinition.id` for references to `ApplicationRecordElement`** ⌈For all `TlvDataIdDefinition` that are referencing `ApplicationRecordElement`s of a given `ApplicationDataType` in the role `tlvRecordElement` the attribute `TlvDataIdDefinition.id` **shall exist and have a unique value** in the context of respective enclosing `ApplicationRecordDataType`.⌋*()*

[constr_1598]{DRAFT} **Scope of the uniqueness of the value of `TlvDataIdDefinition.id` for references to `CppImplementationDataTypeElement`** ⌈For all `TlvDataIdDefinition` that are referencing `CppImplementationDataTypeElements` of a given `CppImplementationDataType`/`CppImplementationDataTypeElement` in the role `tlvSubElement` the attribute `TlvDataIdDefinition.id` **shall exist and have a unique value** in the context of respective enclosing `CppImplementationDataType` or `CppImplementationDataTypeElement`.⌋*()*

Obviously, it is necessary to avoid ambiguity with respect to the definition of TLV data ids. Each model element that can be assigned such an id shall only be assigned one id.

[constr_1599]{DRAFT} **`TlvDataIdDefinition` referencing `ArgumentDataPrototype`** ⌈Each `ArgumentDataPrototype` shall be referenced **at most once** in the role `tlvArgument` in the context of the same `TransformationPropsToServiceInterfaceElementMapping`.⌋*()*

[constr_1600]{DRAFT} **`TlvDataIdDefinition` referencing `ApplicationRecordElement`** ⌈Each `ApplicationRecordElement` shall be referenced **at most once** in the role `tlvRecordElement` in the context of the same `TransformationPropsToServiceInterfaceElementMapping`.⌋*()*

[constr_1601]{DRAFT} **`TlvDataIdDefinition` referencing `CppImplementationDataTypeElement`** ⌈Each `CppImplementationDataTypeElement` shall be referenced **at most once** in the role `tlvSubElement` in the context of the same `TransformationPropsToServiceInterfaceElementMapping`.⌋*()*

[constr_1748]{DRAFT} **Existence of references `TlvDataIdDefinition.tlvArgument`, `TlvDataIdDefinition.tlvRecordElement`, and `TlvDataIdDefinition.tlvSubElement`** ⌈For each `TlvDataIdDefinition`, only one out of the following references shall exist:

- reference to `ArgumentDataPrototype` in the role `tlvArgument`
- reference to `ApplicationRecordElement` in the role `tlvRecordElement`
- reference to `CppImplementationDataTypeElement` in the role `tlvSubElement`.

⌋*()*

[constr_1628]{DRAFT} **Definition of static length field sizes in case of TLV usage** ⌈If the aggregation `tlvDataIdDefinition` exists for a given `TransformationPropsToServiceInterfaceElementMapping` then attributes

- `sizeOfArrayLengthField`,
- `sizeOfStringLengthField`,
- `sizeOfStructLengthField`, and
- `sizeOfUnionLengthField`

shall have a value greater than 0.⌋*()*

Rationale for the existence of [constr_1628]: The TLV serialization requires the usage of length fields:

- If `wire type` 4 is used (for more details, please refer to [TPS_MANI_01186]) then the length field size shall be statically configured.

- If wire types 5-7 are used (see [TPS_MANI_01186]) then the static configuration of the length field size shall also be present since not all length fields are preceded by a tag, e.g. structures contained in an array or the top-level structure contained in a SOME/IP event.

Without demanding the existence of length fields in such a case the result of a serialization could be ambiguous, i.e. make it impossible for the de-serializer to figure out the data layout[7].

[constr_1629]{DRAFT} **Identical sizes of length fields in case of TLV usage** ⌈If the aggregation tlvDataIdDefinition exists for a given TransformationPropsToServiceInterfaceElementMapping then attributes

- sizeOfArrayLengthField,
- sizeOfStringLengthField,
- sizeOfStructLengthField, and
- sizeOfUnionLengthField

shall have an identical value.⌋*()*

Rationale for the existence of [constr_1629]: if wire type 4 is used (for more details, please refer to [TPS_MANI_01186]) and if the receiver encounters a member of a structure or an argument with an unknown tag the de-serializer cannot determine the actual data type of the member of the structure or argument.

[constr_1630]{DRAFT} **No definition of length field sizes on DataPrototype level in case of TLV usage** ⌈If the reference in the role tlvDataIdDefinition exists for a given TransformationPropsToServiceInterfaceElementMapping then attributes

- sizeOfArrayLengthField,
- sizeOfStringLengthField,
- sizeOfStructLengthField, and
- sizeOfUnionLengthField

shall not be individually defined on the level of a DataPrototype (i.e. by means of the reference SomeipDataPrototypeTransformationProps.someipTransformationProps) but only on the level of a ServiceInterface (i.e. by means of the reference TransformationPropsToServiceInterfaceElementMapping.transformationProps).⌋*()*

Rationale for the existence of [constr_1630]: if wire type 4 is used (for more details, please refer to [TPS_MANI_01186]) and if the receiver encounters a member or argument with an unknown tag the de-serializer needs to know the size of the length field.

[7] If a structure consists only of optional elements, it would be hard to detect the case where an array element carries such a structure that happens to set all elements to non-available.

The most reliable way to achieve this is to demand the definition of the size of the length field on the level of the ServiceInterface.

3.18.3.2 Assignment of Wire Type Selection

The TLV encoding supports the definition of a so-called wire type that controls how the information about the length of length fields shall be interpreted.

The meaning of specific settings of the wire type is defined in [7, PRS SOME/IP Protocol].

[TPS_MANI_01186]{DRAFT} **Definition of the applicable wire type** ⌈Attribute ApSomeipTransformationProps.isDynamicLengthFieldSize shall be used to define the applicable wire type.

If the value of attribute ApSomeipTransformationProps.isDynamicLength-FieldSize is set to True then wire type 5-7 shall be used.

If the value of attribute ApSomeipTransformationProps.isDynamicLength-FieldSize does not exist or is set to False then wire type 4 shall be used.⌋ *(RS_MANI_00030)*

3.19 Process Design

Within the definition of e.g. a diagnostic mapping, the assignment to the Process is typically done in a methodological step[8] that happens when all the diagnostic mapping[9] is already complete.

Therefore, it would be good to implement a proxy for an actual Process that can stand in as the target of the relation to a Process at design time. This semantics is realized by meta-class ProcessDesign.

[TPS_MANI_01228]{DRAFT} **Semantics of meta-class ProcessDesign** ⌈Meta-class ProcessDesign shall be used whenever a design-time representation is required for a Process that is designed in a **later** step in the workflow as part of the deployment specification.⌋ *()*

The integrator would have to take care that an actual Process refers to the corresponding ProcessDesign such that by means of this reference an AUTOSAR software tool is able to figure out the relation between a diagnostic mapping and a process, provided that each ProcessDesign is **only** referenced by a single Process.

[constr_1550]{DRAFT} **Reference from Process to ProcessDesign** ⌈Each ProcessDesign shall only be referenced from a single Process.⌋ *()*

[8] i.e. during the creation of the execution manifest

[9] From the methodological point of view, the creation of the diagnostic mapping is typically considered a design-time activity.

Note that the reference from the `Process` to the `ProcessDesign` acknowledges the fact that the `Process` is typically created later in time[10].

Class	**ProcessDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign			
Note	This meta-class has the ability to stand in for a Process at the time when the Process does not yet exist. But its future existence already needs to be considered during design phase and for that a dedicated model element is required.. **Tags:** atp.Status=draft atp.recommendedPackage=ProcessDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
deterministicClientResourceNeeds	DeterministicClientResourceNeeds	*	aggr	This aggregation represents the collection of applicable resource needs for the design of deterministic clients. **Tags:**atp.Status=draft
executable	Executable	0..1	ref	Reference to executable that is executed in the process. **Tags:**atp.Status=draft

Table 3.146: ProcessDesign

Conceivably, the association of diagnostic mappings with Meta-class `ProcessDesign` may still happen as a finalizing last step of the activity to create the diagnostic mappings. To accommodate for this potential modeling, the reference from a diagnostic mapping to `ProcessDesign` has been decorated by stereotype ≪`atpSplitable`≫.

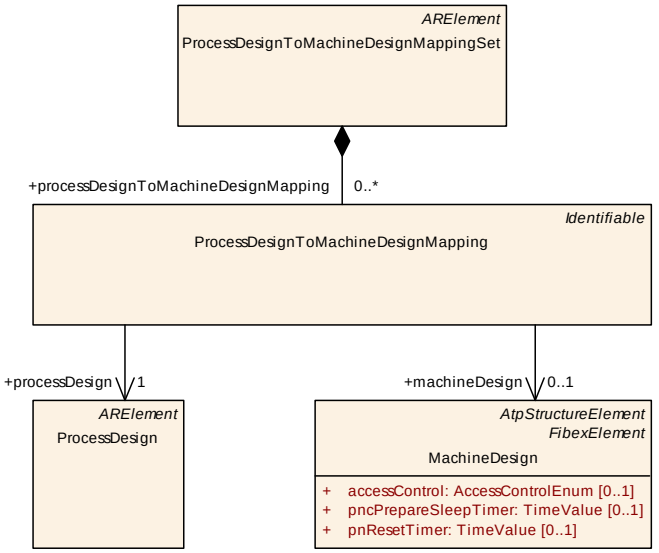


Figure 3.75: Modeling of the `ProcessDesignToMachineDesignMapping`

For more information concerning the semantics of this stereotype please refer to the specification of the AUTOSAR Generic Structure Template [6].

[10] In other words, if references are needed between design-related and deployment-related meta-classes then the direction of these references shall always point from deployment to design.

[constr_1693]{DRAFT} **Relation of `Executable`, `ProcessDesign`, and `Process`** ⌈Any `Executable` that is referenced by a `ProcessDesign` shall also be referenced by every `Process` that references the `ProcessDesign`.⌋*()*

[TPS_MANI_01229]{DRAFT} **Pre-allocation of a given `ProcessDesign` on a specific `MachineDesign`** ⌈It is also possible to pre-allocate a given `ProcessDesign` on a specific `MachineDesign`. For this purpose meta-class `ProcessDesignToMachineDesignMapping` exists.⌋*()*

The semantics of meta-class `MachineDesign` is explained in section 5.

Class	**ProcessDesignToMachineDesignMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This element is used in the design phase to predefine a mapping of a process to a machine. Such a mapping may be overruled in the deployment phase. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
machineDesign	MachineDesign	0..1	ref	This reference identifies the MachineDesign in the context of the ProcessDesignToMachineDesignMapping. **Tags:**atp.Status=draft
processDesign	ProcessDesign	1	ref	This reference identifies the ProcessDesign in the context of the ProcessDesignToMachineDesignMapping. **Tags:**atp.Status=draft

Table 3.147: ProcessDesignToMachineDesignMapping

Please note that an intended `ProcessDesignToMachineDesignMapping` may not be possible for utilization of the target machine and therefore a different `ProcessToMachineMapping` may be created in the deployment phase.

3.19.1 Deterministic Client Resource

Meta-class `ProcessDesign` can also be used to add support for the so-called `Deterministic Client`.

Please note that an explanation of the specific meaning of the term `Deterministic Client` is out of the scope of this document. A detailed explanation can be found in the SWS Execution Management [18].

To formalize the support for the `Deterministic Client`, meta-class `DeterministicClientResourceNeeds` is aggregated at `ProcessDesign`.

Class	**DeterministicClientResourceNeeds**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign			
Note	This meta-class specifies process and cycle specific computing resource needs of DeterministicClient library functions. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
hardware Platform	String	0..1	attr	This attribute represents a textual identification of the target platform.
initResource	DeterministicClient Resource	0..1	aggr	This represents the computing resource needs of a DeterministicClient::WaitForNextActivation kInit cycle. **Tags:**atp.Status=draft
runResource	DeterministicClient Resource	0..1	aggr	This represents the computing resource needs of a DeterministicClient::WaitForNextActivation kRun cycle. **Tags:**atp.Status=draft

Table 3.148: DeterministicClientResourceNeeds

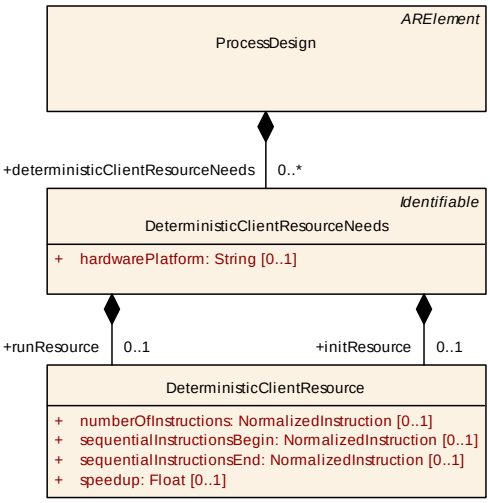


Figure 3.76: Modeling of the `DeterministicClientResourceNeeds`

[TPS_MANI_01199]{DRAFT} **Semantics of `DeterministicClientResourceNeeds`** ⌈Meta-class `DeterministicClientResourceNeeds` aggregates `DeterministicClientResource` in two roles in order to be able to specify resource needs in two different contexts of the execution of a `Deterministic Client`.⌋*(RS_MANI_00050)*

Class	**DeterministicClientResource**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign
Note	This meta-class specifies computing resource needs of DeterministicClient library functions. **Tags:**atp.Status=draft
Base	*ARObject*

▽

△

Class	**DeterministicClientResource**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
numberOf Instructions	NormalizedInstruction	0..1	attr	This attribute represents the normalized runtime consumption on the target system within one DeterministicClient::WaitForNextActivation cycle, assuming the "worst-case" runtime where the workers would be executed sequentially.
sequential Instructions Begin	NormalizedInstruction	0..1	attr	Normalized sequential runtime at the beginning of the DeterministicClient::WaitForNextActivation cycle (which mostly cannot be parallelized), before the main usage of the worker pool starts.
sequential InstructionsEnd	NormalizedInstruction	0..1	attr	WaitForNextActivation cycle (which mostly cannot be parallelized), after the main usage of the worker pool has ended.
speedup	Float	0..1	attr	This attribute defines how much faster the calculations within one DeterministicClient::WaitForNextActivation cycle can be finished if numberOfWorkers are physically available, i.e. if enough cores were available on the machine to perform parallel execution of all workers (sequential runtime / parallelized runtime).

Table 3.149: DeterministicClientResource

[TPS_MANI_01200]{DRAFT} **Semantics of meta-class `DeterministicClientResource`** ⌈Meta-class `DeterministicClientResource` defines several attributes that provide information about the nature of the execution of worker threads. The values of these attributes are given a dimensionless `NormalizedInstruction`.

Nevertheless, the values of the attributes

- `numberOfInstructions`
- `sequentialInstructionsBegin`
- `sequentialInstructionsEnd`

are only valid for a specific hardware platform. The purpose of using `NormalizedInstruction` is to align resource usage of different `Process`es (possibly from different vendors) at integration time.⌋*(RS_MANI_00050)*

Primitive	**NormalizedInstruction**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign
Note	This meta-class is used to describe runtime budget needs on the target system within Deterministic Client::WaitForNextActivation cycles. NormalizedInstructions does not reflect the actual number of code instructions, but allows the description of comparative resource needs. NormalizedInstructions is used for configuration of computing resources at integration time. NormalizedInstruction = runtime in sec * clock frequency in Hz **Tags:** atp.Status=draft xml.xsd.customType=NORMALIZED-INSTRUCTION xml.xsd.pattern=[1-9][0-9]* xml.xsd.type=string

Table 3.150: NormalizedInstruction

3.20 Grant Design

The definition of intents (for example: `ClientComSpec.clientIntent`) as described in chapter 3.15.5.1.3 is used to express the intention of the software designer to use (or refrain from using) specific APIs in the application software.

The definition of intents represents one aspect of the identity and Access Management (IAM). Another aspect of the IAM configuration is the definition of the actual permissions granted by the platform software.

The modeling of such grants is done on two levels:

- the definition of `GrantDesign` allows for the pre-specification of grants already on the design level. The modeling of `GrantDesign` is described in this chapter.
- the definition of `Grant` allows for the actual and final specification of grants from the perspective of the platform software. The modeling of Grant is described in chapter 9.9.

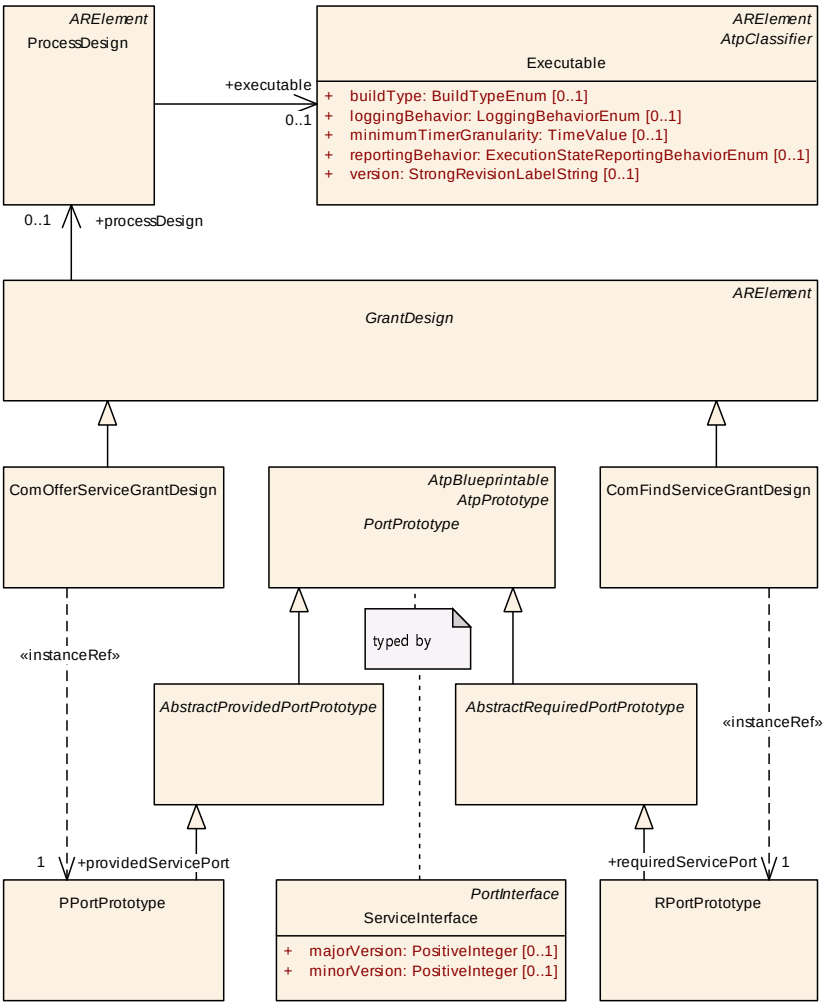


Figure 3.77: Modeling of grant designs for service discovery

Class	**GrantDesign** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign			
Note	This meta-class serves as an abstract base class for the description of grants on design level. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	ComFindServiceGrantDesign, *ComGrantDesign*, ComOfferServiceGrantDesign, RawDataStreamGrant Design			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
processDesign	ProcessDesign	0..1	ref	This reference identifies the corresponding Process Design that gives context to the GrantDesing. **Tags:**atp.Status=draft

Table 3.151: GrantDesign

Abstract meta-class `GrantDesign` acts as the base class for the definition of grants on the design level.

Grants are specific for a given `Process`. In other words, two `Process`es created from the same `Executable` may be assigned different sets of grants. This specific relation shall also be available on the design level.

[TPS_MANI_01231]{DRAFT} **`GrantDesign` references `ProcessDesign`** ⌈Meta-class `GrantDesign` references `ProcessDesign` as a means to design the set of `Grant`s for the given `Process`.⌋*(RS_MANI_00060)*

3.20.1 Com Grant Design

Subclasses of `GrantDesign` are created to cover specific aspects of grants for communication on the *AUTOSAR adaptive Platform*.

[TPS_MANI_01232]{DRAFT} **Semantics of meta-class `ComOfferServiceGrantDesign`** ⌈The existence of a `ComOfferServiceGrantDesign` that references a specific `AbstractProvidedPortPrototype` in the role `providedServicePort` indicates that the design foresees that the referenced `AbstractProvidedPortPrototype` shall be granted rights to offer the respective service.⌋*(RS_MANI_00060)*

Please note that there is no explicitly modeled intent that corresponds to the existence of the `ComOfferServiceGrantDesign`. The understanding is that the mere existence of an `AbstractProvidedPortPrototype` typed by a `ServiceInterface` indicates the intent to offer a service.

[TPS_MANI_01233]{DRAFT} **Semantics of meta-class `ComFindServiceGrantDesign`** ⌈The existence of a `ComFindServiceGrantDesign` that references a specific `AbstractRequiredPortPrototype` in the role `requiredServicePort` indicates that the design foresees that the referenced `AbstractRequiredPortPrototype` shall be granted rights to find the respective service.⌋*(RS_MANI_00060)*

Please note that there is no explicitly modeled intent that corresponds to the existence of the ComFindServiceGrantDesign.

The understanding is that the mere existence of an AbstractRequiredPortPrototype typed by a ServiceInterface indicates the intent to find a service.

Class	**ComOfferServiceGrantDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class represents the ability to define a Grant for offering a service. **Tags:** atp.Status=draft atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
providedServicePort	PPortPrototype	1	iref	This instanceRef identifies the PPortPrototype on which the service shall be offered. **Tags:**atp.Status=draft **InstanceRef implemented by:**PPortPrototypeInExecutableInstanceRef

Table 3.152: ComOfferServiceGrantDesign

Class	**ComFindServiceGrantDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class represents the ability to define a Grant for finding a service. **Tags:** atp.Status=draft atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
requiredServicePort	RPortPrototype	1	iref	This instanceRef identifies the RPortPrototype on which the service shall be found. **Tags:**atp.Status=draft **InstanceRef implemented by:**RPortPrototypeInExecutableInstanceRef

Table 3.153: ComFindServiceGrantDesign

[TPS_MANI_01234]{DRAFT} **Semantics of ComFieldGrantDesign** ⌈The existence of a ComFieldGrantDesign that references a specific Field in the role field indicates that the design foresees that the application software shall be granted rights to access the respective Field. The nature of the access, i.e. get vs. set is specified by means of the attribute role.⌋*(RS_MANI_00060)*

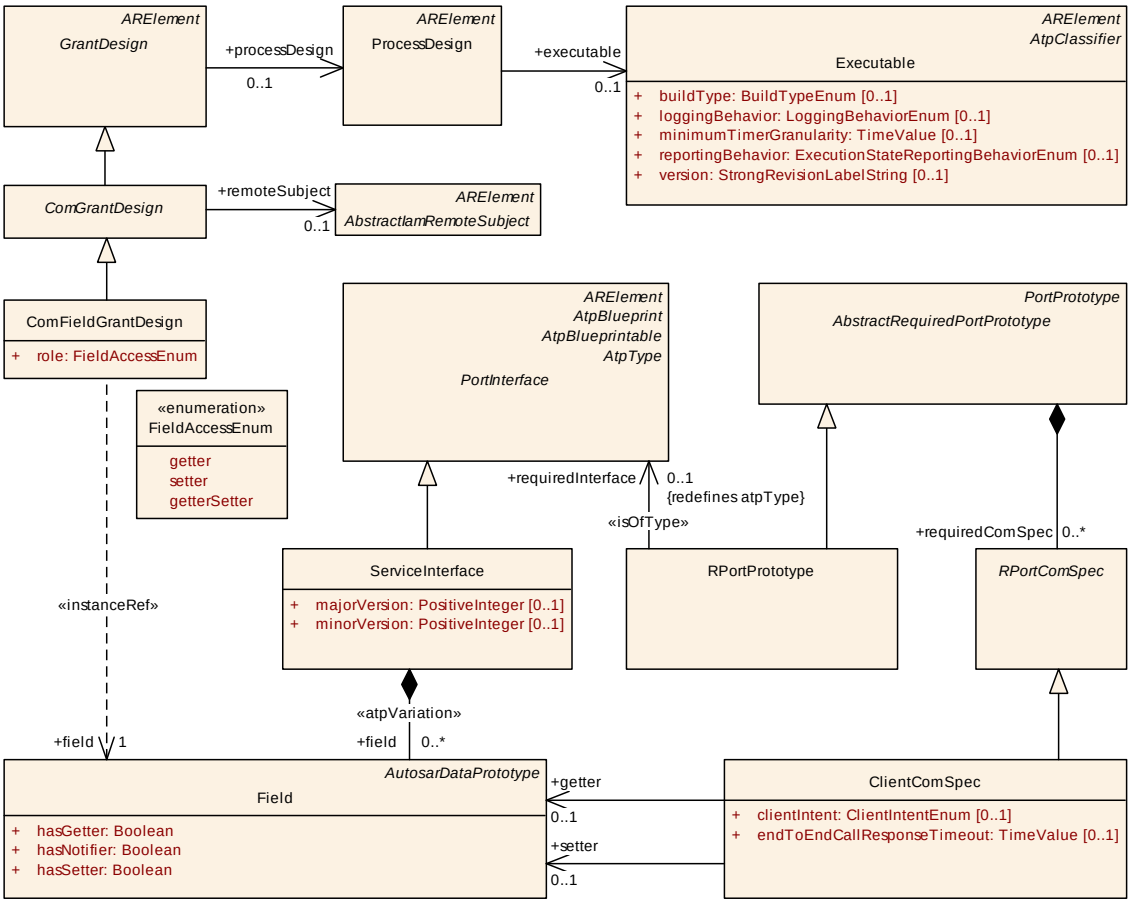


Figure 3.78: Modeling of grant designs for `field`

Class	**ComFieldGrantDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class represents the ability to define a Grant for a ServiceInterface.field. **Tags:** atp.Status=draft atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrantDesign*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
field	Field	1	iref	Reference to the affected Field in the context of an Executable. **Tags:**atp.Status=draft **InstanceRef implemented by:**FieldInExecutableInstanceRef
role	FieldAccessEnum	1	attr	This attribute provides the ability to further specify the access to the ServiceInterface.field from a design perspective.

Table 3.154: ComFieldGrantDesign

Enumeration	**FieldAccessEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant
Note	This meta-class provides values that qualify access to a field. **Tags:**atp.Status=draft
Literal	***Description***
getter	Access to the getter of the Field. **Tags:**atp.EnumerationLiteralIndex=0
getterSetter	Access to getter and setter of the field **Tags:**atp.EnumerationLiteralIndex=2
setter	Access to the setter of the Field. **Tags:**atp.EnumerationLiteralIndex=1

Table 3.155: FieldAccessEnum

[TPS_MANI_01235]{DRAFT} **Semantics of `ComEventGrantDesign`** ⌈The existence of a `ComEventGrantDesign` that references a specific `VariableDataPrototype` that is aggregated in the role `event` by the enclosing `ServiceInterface` indicates that the design foresees that the application software shall be granted rights to access the respective `event`.⌋*(RS_MANI_00060)*

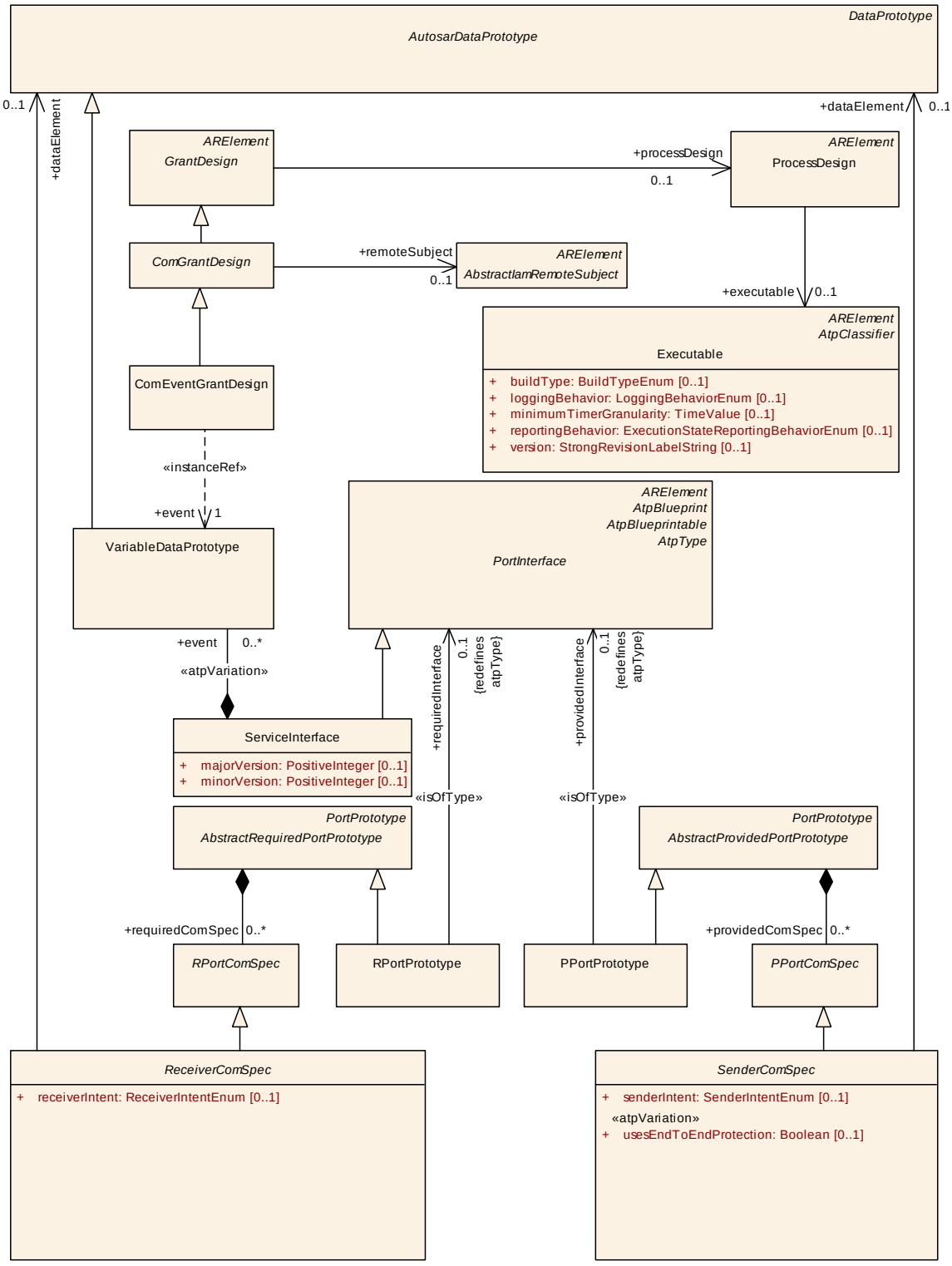


Figure 3.79: Modeling of grant designs for `event`

Class	ComEventGrantDesign			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class represents the ability to define a Grant for a ServiceInterface.event. **Tags:** atp.Status=draft atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrantDesign*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
event	VariableDataPrototype	1	iref	This reference represents the affected event. **Tags:**atp.Status=draft **InstanceRef implemented by:**EventInExecutable InstanceRef

Table 3.156: ComEventGrantDesign

[TPS_MANI_01236]{DRAFT} **Semantics of `ComMethodGrantDesign`** ⌈The existence of a `ComMethodGrantDesign` that references a specific `ClientServerOperation` that is aggregated in the role `method` by the enclosing `ServiceInterface` indicates that the design foresees that the application software shall be granted rights to call the respective `method`.⌋*()*

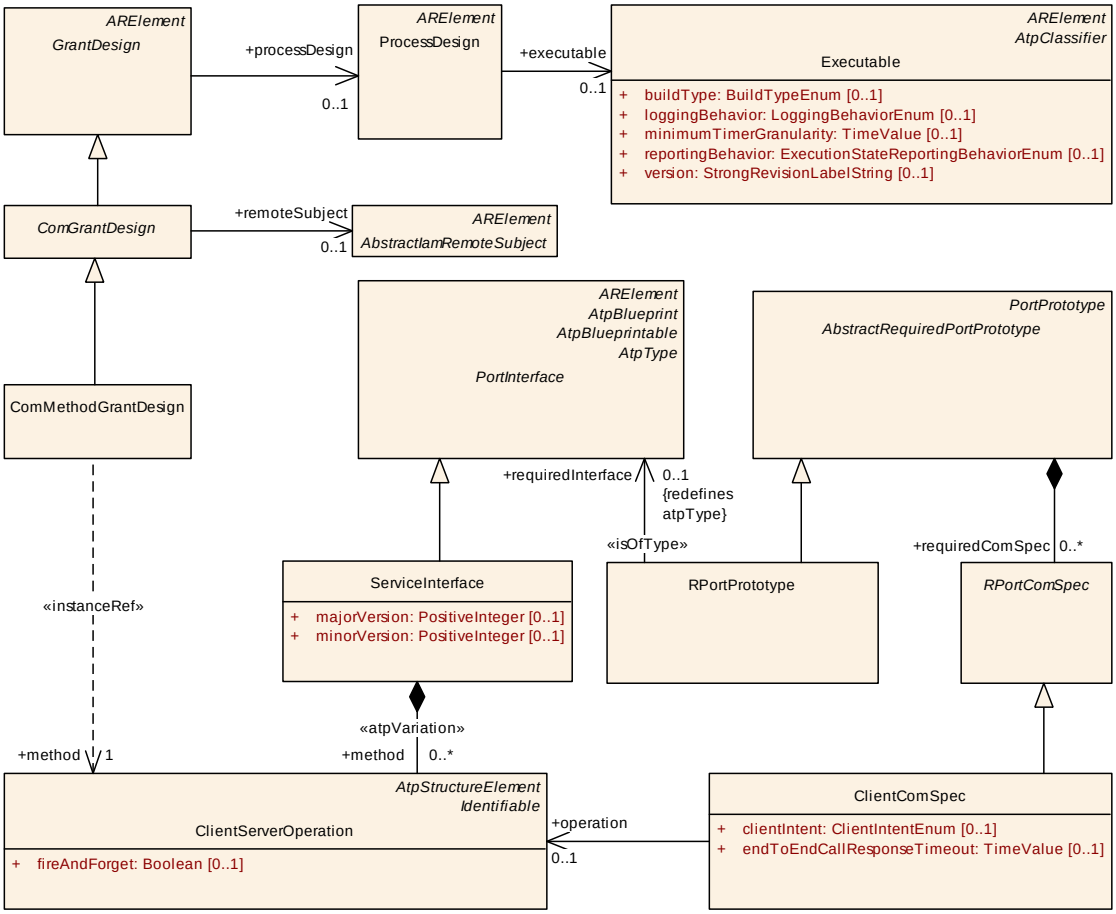


Figure 3.80: Modeling of grant designs for `method`

Class	**ComMethodGrantDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class represents the ability to define a Grant for a ServiceInterface.method. **Tags:** atp.Status=draft atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrantDesign*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
method	ClientServerOperation	1	iref	This reference identifies the corresponding method. **Tags:**atp.Status=draft **InstanceRef implemented by:**RequiredMethodIn ExecutableInstanceRef

Table 3.157: ComMethodGrantDesign

3.20.2 Grant Design for Raw Streaming Data

The usage of a raw data stream is subject to restrictions imposed by the IAM. Therefore, meta-class `RawDataStreamGrantDesign` exists to support this use case.

[TPS_MANI_01284]{DRAFT} **Granularity of meta-class `RawDataStreamGrantDesign`** ⌈The granularity of the `RawDataStreamGrantDesign` is the entire `AbstractRawDataStreamInterface`. It is not expected that a definition of an IAM policy makes sense on a smaller level, i.e. on the level of `ClientServerOperation` aggregated by a `AbstractRawDataStreamInterface`.⌋*()*

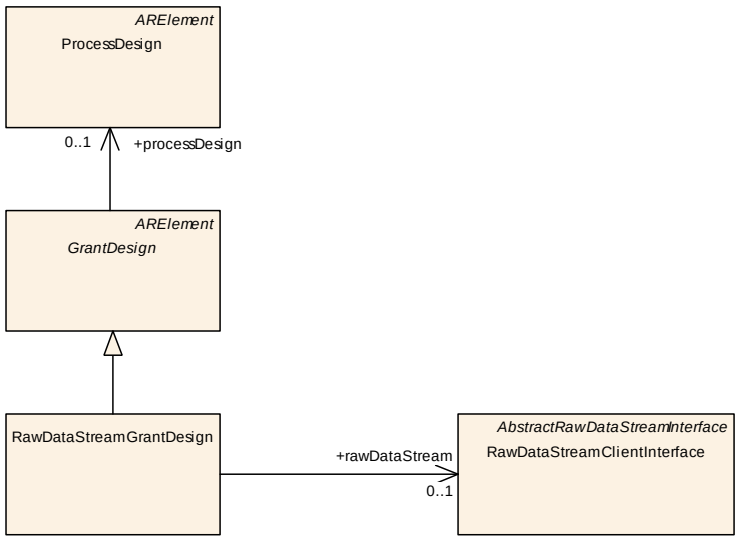


Figure 3.81: Modeling of the `RawDataStreamGrantDesign`

Class	**RawDataStreamGrantDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::RawDataStreamGrant			
Note	This meta-class represents the ability to define the IAM configuration for a RawDataStream on design level. **Tags:** atp.Status=draft atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
rawDataStream	RawDataStreamClient Interface	0..1	ref	This reference identifies the applicable RawDataStream Interface. **Tags:**atp.Status=draft

Table 3.158: RawDataStreamGrantDesign

3.20.3 Remote access control

The definition of the deployment for the *Identity and Access Manager* and the definition of grants relies on the local enforcement of identity and access permissions. In other words it is possible for example to define that a particular `method` of a `ServiceInterface` is allowed to be called by a local `Process` on the local `Machine`. But it is not possible to restrict the remote `Machine`s that are allowed to call this `method`.

The fact that the `Machine` on which the service is running has no mean to make additional checks on the incoming requests enables processing of wrongly issued requests by a healthy remote `Machine` as well as escalation of privileges by an attacker via issuing arbitrary request towards services from a compromised remote `Machine`.

Most of the times it is not possible for a `Machine` to recognize that its communicating peer is compromised because the attacker has access to all the resources of that `Machine` and can run in stealth mode. An effective way to minimize the damage of a compromised remote `Machine` is to enforce additional checks on the incoming requests at the receiver side ensuring that remote `Machine` cannot go beyond what they could request in a healthy state.

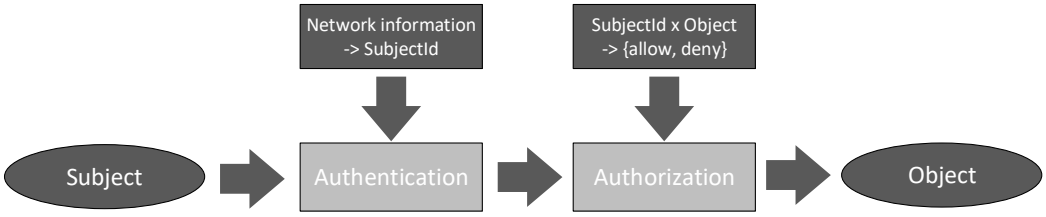


Figure 3.82: Access policy enforcement based on the Subject ID from the network binding

The access control process aims at enforcing policies on the relation between a “Subject” and an “Object”. In the example where an remote Machine makes calls to a service interface, the remote Machine is the Subject and the method of a ServiceInterface is the Object.

The access control process comprises of the two main operations, namely, Authentication and Authorization, which are mostly independent. During the authentication process the identity of the subject is verified and an authentic identifier is resolved. Authentication is an essential part of the chain to ensure that different subjects cannot impersonate each other.

In the next step, during Authorization, the identity of the Subject is checked upon the rules and policies defined for the accessing the Object to verify if the Subject’s request is legitimate. These policies shall be defined by the system or the resource owner.

The authentication of the remote subject is based on the network binding. When a secure channel is established, the remote peer has already gone through an authentication protocol. Therefore, the identity information can be forwarded to the IAM to apply the corresponding defined policies that are defined for the requests coming from that channel as depicted in Figure 3.82.

The remote subject is modeled as a specialization of AbstractIamRemoteSubject. The different specializations will be presented in the following sections.

Class	***AbstractIamRemoteSubject*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM			
Note	This abstract meta-class defines the proxy information about the remote node. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	IPSecIamRemoteSubject, IpIamRemoteSubject, TlsIamRemoteSubject			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.159: AbstractIamRemoteSubject

With the modeling of ComGrantDesigns the permissions that are granted by the platform software are defined. As an option a ComGrantDesign is able to reference the AbstractIamRemoteSubject in the role remoteSubject.

[TPS_MANI_03238]{DRAFT} **Definition of ComMethodGrantDesign.remoteSubject** ⌈If the ComMethodGrantDesign references one or several AbstractIamRemoteSubjects in the role remoteSubject then the design foresees that only the defined remoteSubjects shall be granted rights to access the ClientServerOperation that is referenced in the role method by the same ComMethodGrantDesign.⌋*()*

[TPS_MANI_03239]{DRAFT} **Definition of ComEventGrantDesign.remoteSubject** ⌈If the ComEventGrantDesign references one or several AbstractIamRemoteSubjects in the role remoteSubject then the design foresees that only the

defined remoteSubjects shall be granted rights to access the VariableDataPrototype that is referenced in the role event by the same ComEventGrantDesign.⌋ ()

[TPS_MANI_03251]{DRAFT} **Definition of ComFieldGrantDesign.remoteSubject** ⌈If the ComFieldGrantDesign references one or several AbstractIamRemoteSubjects in the role remoteSubject then the design foresees that only the defined remoteSubjects shall be granted rights to access the Field that is referenced in the role field by the same ComFieldGrantDesign.⌋()

3.20.3.1 Remote subject in case of TLS

This chapter defines how a AbstractIamRemoteSubject is modeled in case of a TLS-based secure channel.

[TPS_MANI_03240]{DRAFT} **Modeling of a remote peer in case of TLS-based secure channel** ⌈In case of TLS-based secure channel the remote peer is modeled as TlsIamRemoteSubject that is identified either by

- a CryptoServiceCertificate that is referenced by the TlsIamRemoteSubject in the role acceptedRemoteCertificate,
- a Pre-shared Key that is referenced by the TlsIamRemoteSubject via TlsCryptoCipherSuite in the role acceptedCryptoCipherSuiteWithPsk.

⌋()

Please note that the security of a pre-shared key as authentication in TLS protocol depends on the number of entities sharing the same key. If multiple Machines are using the same shared key, one cannot reliably distinguish between those Machines because any of them can impersonate the others.

It can only be ensured that no other Machine without the knowledge of the pre-shared key can established a secure channel.

The TlsIamRemoteSubject may be identified by using certificates in two ways. First, it is possible to directly specify the certificates that the TlsIamRemoteSubject may to use by referencing them and setting derivedCertificateAccepted to false.

This approach requires the presence of the remote certificate on the local Machine. Secondly, by setting derivedCertificateAccepted to true it is possible to specify the Common Name (as given in the X509 Certificate) of the TlsIamRemoteSubject.

In that case, the acceptedRemoteCertificates define the set of allowed root certificates for the certificate presented by the TlsIamRemoteSubject.

The reason for the upper multiplicity is that the OEM may have multiple suppliers for a Machine and it shall be allowed to define that in such a case all these Machines are allowed to connect even though they have different certificate chains.

[TPS_MANI_03241]{DRAFT} **Modeling of relevant `TlsSecureComProps` for `TlsIamRemoteSubject`** ⌈With the `TlsIamRemoteSubject.iamRelevantTlsSecureComProps` reference it is possible to define all `TlsSecureComProps` that the `TlsIamRemoteSubject` supports to establish a secure channel.⌋()

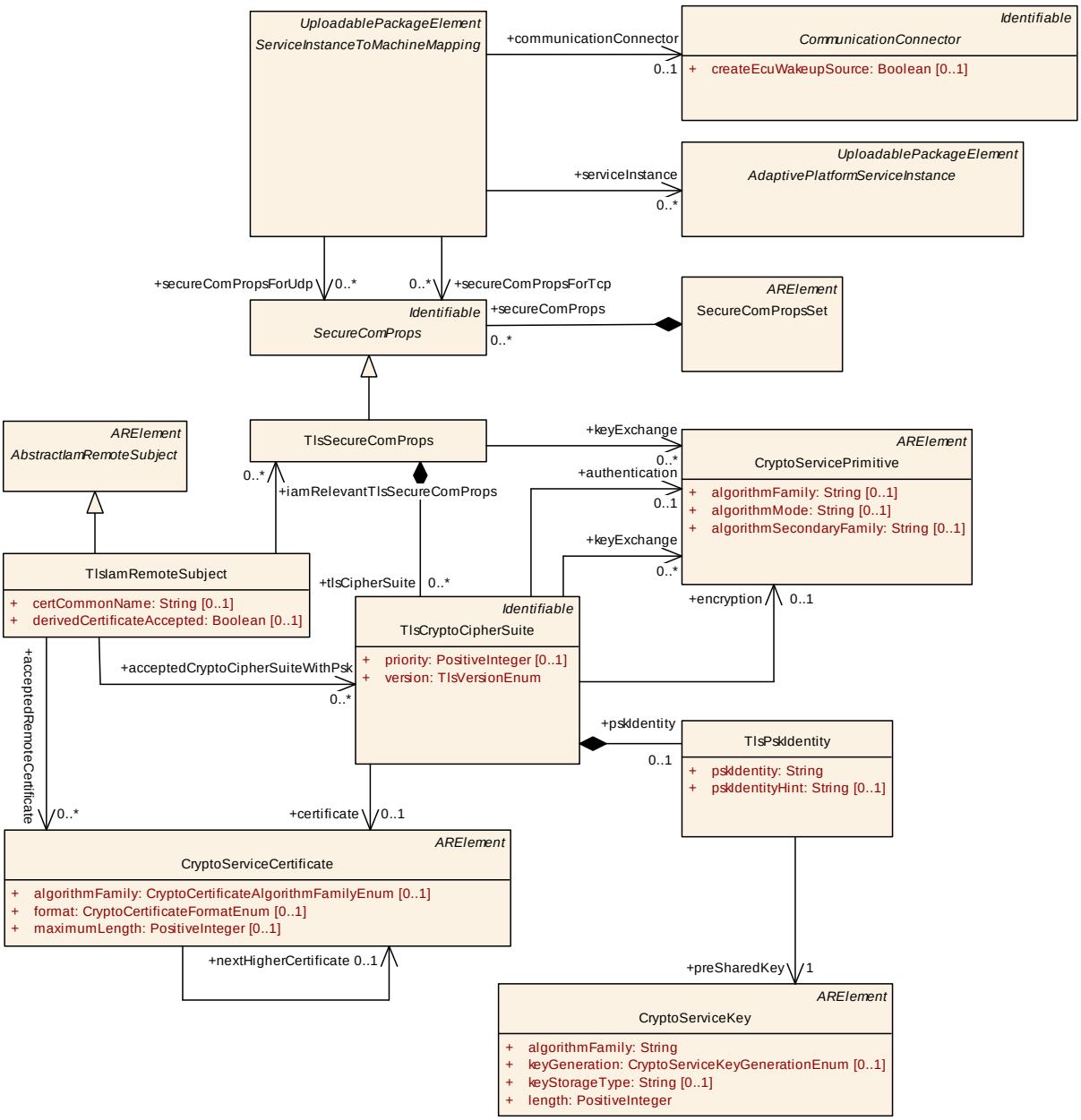


Figure 3.83: Proxy information about the remote node in case of TLS

Class	TlsIamRemoteSubject			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM			
Note	This meta-class defines the proxy information about the remote node in case of TLS. **Tags:** atp.Status=draft atp.recommendedPackage=IamRemoteSubjects			
Base	*ARElement*, *ARObject*, *AbstractIamRemoteSubject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
acceptedCryptoCipherSuiteWithPsk	TlsCryptoCipherSuite	*	ref	This reference is used to identify a remote node by means of the preshared Key. **Tags:**atp.Status=draft
acceptedRemoteCertificate	CryptoServiceCertificate	*	ref	This reference is used to identify a remote node by means of the certificate. **Tags:**atp.Status=draft
certCommonName	String	0..1	attr	This attribute defines the common name (CN) of the certificate of the remote peer.
derivedCertificateAccepted	Boolean	0..1	attr	This attribute defines whether a derivedCertificate is accepted (true) or not (false).
iamRelevantTlsSecureComProps	TlsSecureComProps	*	ref	This reference defines the local TlsSecureComProps that are relevant for IAM. **Tags:**atp.Status=draft

Table 3.160: TlsIamRemoteSubject

3.20.3.2 Remote subject in case of IPsec

This chapter defines how a `AbstractIamRemoteSubject` is modeled in case of a IPsec-based secure channel.

[TPS_MANI_03242]{DRAFT} **Modeling of a remote peer in case of IPsec-based secure channel** ⌈In case of IPsec-based secure channel the remote peer is modeled as `IPSecIamRemoteSubject` that is identified by `IPSecRule`s that are referenced by `localIpSecRule`. The `IPSecRule`s define all secure connections that the remote peer is allowed to establish.⌋*()*

Please note that the local IP Address of the remote peer is defined by the `Network-Endpoint` that aggregates the `IPSecRule`s.

Class	IPSeclamRemoteSubject
Package	M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM
Note	This meta-class defines the proxy information about the remote node in case of IPsec. **Tags:** atp.Status=draft atp.recommendedPackage=IamRemoteSubjects

▽

△

Class	IPSeclamRemoteSubject			
Base	ARElement, ARObject, AbstractIamRemoteSubject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
localIpSecRule	IPSecRule	*	ref	This reference is used to describe theRemoteSubjects local IPSecRules. **Tags:**atp.Status=draft

Table 3.161: IPSeclamRemoteSubject

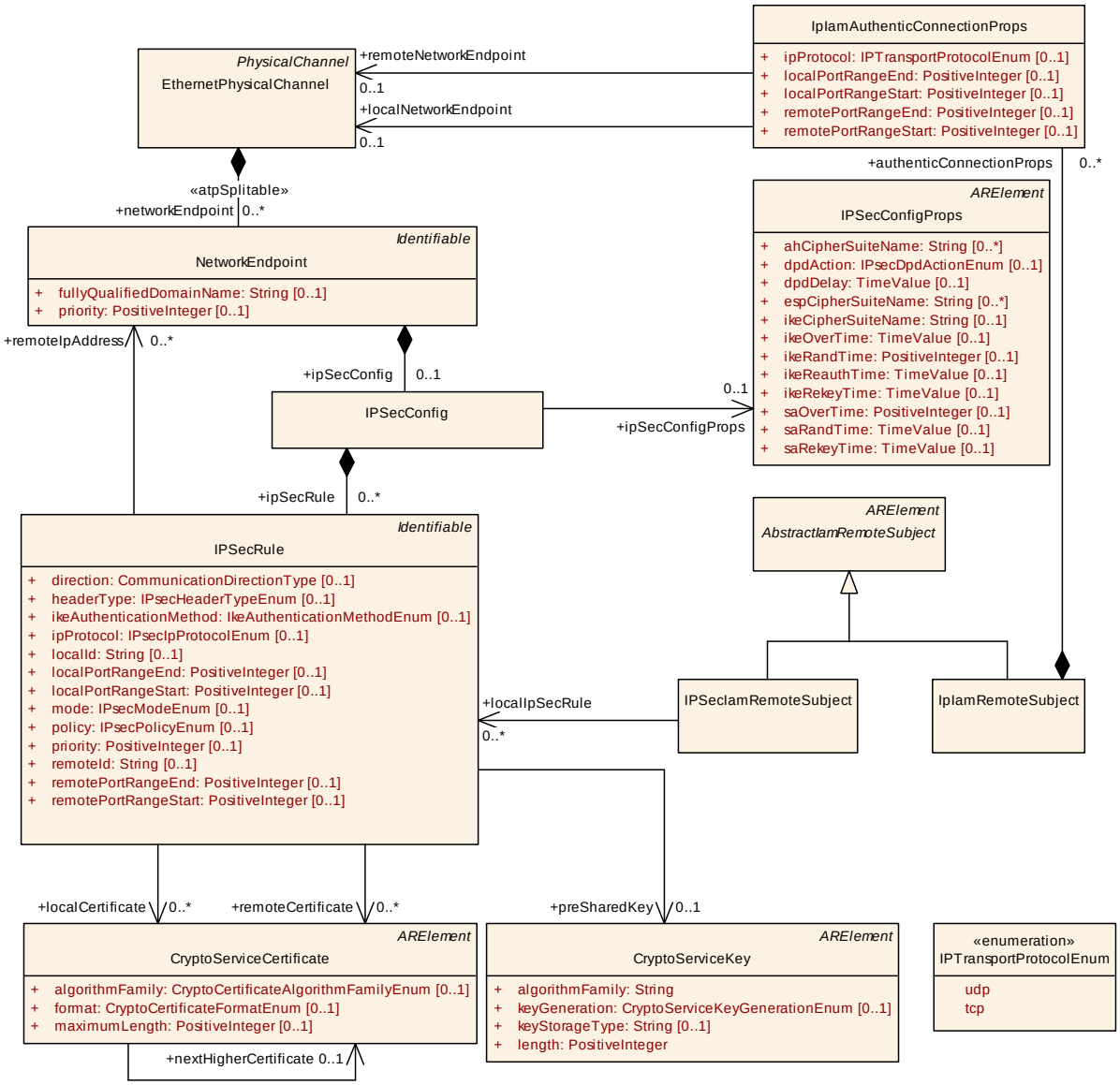


Figure 3.84: Proxy information about the remote node in case of IPsec

3.20.3.3 Remote subject in case of IP communication

Please note that it is possible to define a `AbstractIamRemoteSubject` that is based on the general IP communication. In this case no details about how the communication is secured are given and actually securing the communication (e.g., cryptographically, via hardware mechanism, or appropriate network and switch design) is not part of the model. A `IpIamRemoteSubject` is identified by a combination of a local and a remote IP address, local and remote port ranges, and a transport protocol.

[TPS_MANI_03244]{DRAFT} **Modeling of a remote peer in case of a general IP communication** ⌈In case of a general IP communication the remote peer is modeled as `IpIamRemoteSubject` that is identified by the `NetworkEndpoint` that is referenced by the `localNetworkEndpoint` reference. The defined remote peer is allowed to establish IP connections to the `remoteNetworkEndpoint` over the `ipProtocol` and the defined local port range and remote port range.⌋*()*

Class	**IpIamRemoteSubject**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM			
Note	This meta-class defines the proxy information about the remote node in case of general IP communication. **Tags:** atp.Status=draft atp.recommendedPackage=IamRemoteSubjects			
Base	*ARElement*, *ARObject*, *AbstractIamRemoteSubject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authentic ConnectionProps	IpIamAuthentic ConnectionProps	*	aggr	Definition of IP rules assigned to the IpIamRemote Subject. **Tags:**atp.Status=draft

Table 3.162: IpIamRemoteSubject

Class	**IpIamAuthenticConnectionProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM			
Note	This meta-class defines a set of properties for IP connections in the context of IAM configuration. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ipProtocol	IPTransportProtocol Enum	0..1	attr	This attribute defines the relevant IP protocol.
localNetwork Endpoint	EthernetPhysical Channel	0..1	ref	This reference defines an authentic local Network Endpoint in terms of IAM configuration. **Tags:**atp.Status=draft
localPortRange End	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines an end value for the local port range.
localPortRange Start	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines a start value for the local port range.

▽

△

Class	IplamAuthenticConnectionProps			
remoteNetwork Endpoint	EthernetPhysical Channel	0..1	ref	This reference defines an authentic remote Network Endpoint in terms of IAM configuration. **Tags:**atp.Status=draft
remotePort RangeEnd	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines an end value for the remote port range.
remotePort RangeStart	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines a start value for the remote port range.

Table 3.163: IplamAuthenticConnectionProps

3.20.3.4 Remote subject in case of SecOC communication

The identity information in the case of SecOC depends on the group of `Machine`s that are sharing the same cryptographic key.

In other words, if a valid SecOC message is received with a given key it is given that only remote `Machine`s that “know the key” were able to send the message. The key is associated with a DataId and defines the “object” in the access control model. If a message received for a given DataID cannot be validated, then it will be dropped. Therefore, the access control between the remote subject and local object is taking place.

To summarize, the modeling of a Remote subject in case of SecOC cannot provide additional benefit neither by increasing the granularity of the subject identification nor providing new enforcement of rules on the object.

4 Diagnostic Design

4.1 Diagnostic Mapping

4.1.1 Overview

The configuration of diagnostics on the *AUTOSAR adaptive platform* will typically be done by creating a Diagnostic Extract by means of the Diagnostic Extract Template [19] that is also used on the *AUTOSAR classic platform*.

Therefore, concepts within the Diagnostic Extract should be similarly applicable to models on both platforms uniformly.

It can even be safely expected that a given Diagnostic Extract can be divided into parts that apply for ECUs build on top of the *AUTOSAR classic platform* and parts that apply to ECUs built on top of the *AUTOSAR adaptive platform* that all belong to the same vehicle.

In terms of applicability to this document, the part of the Diagnostic Extract that is relevant in this context is the mapping between the definition of information related to diagnostic protocol content and the application software.

Following the pattern of communication on the *AUTOSAR adaptive platform*, interaction between the application software and platform modules for diagnostics (the so-called `AUTOSAR Adaptive Diagnostic Management`) is also using service-oriented communication.

This raises the question of how the communication ends on both application and platform software get together in the course of a service discovery. This issue can be addressed by utilizing modeling concepts existing in a Diagnostic Extract on the *AUTOSAR adaptive platform*.

Specifically, by formally modeling the relation between the `AUTOSAR Adaptive Diagnostic Management` and specific endpoints in the application software it is possible to configure the service-oriented communication in a way that communication endpoints that are supposed to be connected become actually connected to each other as the service discovery unfolds.

The meta-classes that need to be considered for this purpose are in the following list:

- `DiagnosticEventPortMapping`
- `DiagnosticOperationCyclePortMapping`
- `DiagnosticEnableConditionPortMapping`
- `DiagnosticClearConditionPortMapping`
- `DiagnosticIndicatorPortMapping`
- `DiagnosticMemoryDestinationPortMapping`

- DiagnosticSecurityLevelPortMapping
- DiagnosticServiceDataIdentifierPortMapping

In order to exemplify the approach, the diagram depicted in Figure 4.1 describes a very simplistic situation where two different PPortPrototypes typed by possibly two different DiagnosticDataIdentifierInterface exposed by an AdaptiveApplicationSwComponentType is accessed by the AUTOSAR Adaptive Diagnostic Management on the *AUTOSAR adaptive platform* with the purpose of accessing an entire DID.

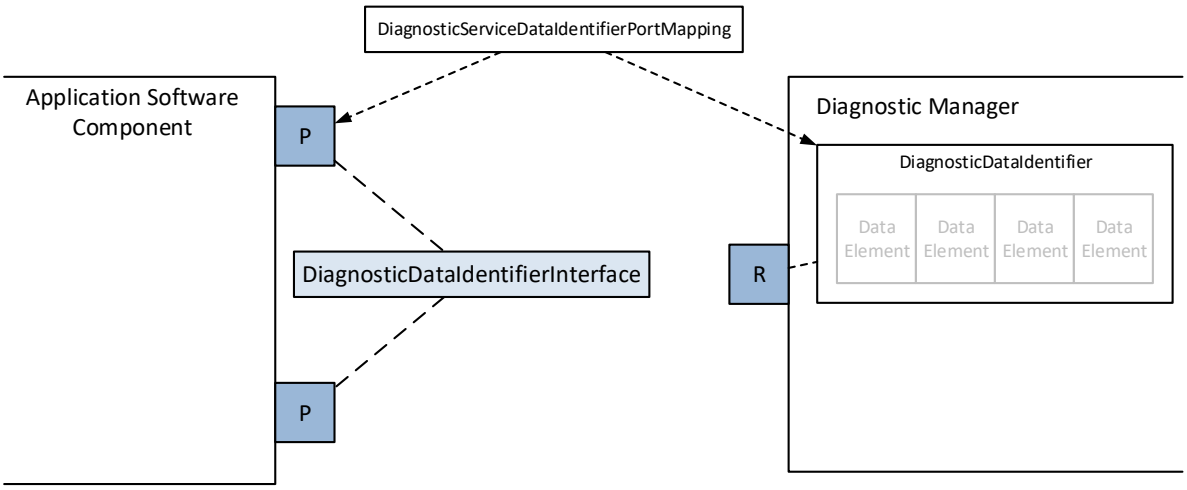


Figure 4.1: Example data exchange for diagnostic purpose

In this situation, the AUTOSAR Adaptive Diagnostic Management obviously needs to be aware which of the two available PPortPrototypes has to be accessed from the depicted RPortPrototype for working with a given DiagnosticDataIdentifier in particular.

If it were possible to identify the matching pairs of PortPrototypes then the communication channel between them could be established either completely or at least to a large extent automatically.

Please note that this statement might or might not involve the execution of a service discovery. In many cases it will not.

From the technical point of view, the AUTOSAR meta-model provides means to achieve the discussed formalization of the relation between an element of the diagnostics configuration (in this case a DiagnosticDataIdentifier that is represented by a PortPrototype on the surface of the DM) and a PortPrototype exposed by the application software.

In particular, a subclass of DiagnosticSwMapping (in this specific case: DiagnosticServiceDataIdentifierPortMapping) formalizes the “connection” between both ends of the communication.

Class	***DiagnosticSwMapping*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::ServiceMapping			
Note	This represents the ability to define a mapping between a diagnostic information (at this point there is no way to become more specific about the semantics) to a software-component.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticClearConditionPortMapping, DiagnosticEnableConditionPortMapping, DiagnosticEventPortMapping, DiagnosticFimFunctionMapping, DiagnosticIndicatorPortMapping, DiagnosticMemoryDestinationPortMapping, DiagnosticOperationCyclePortMapping, DiagnosticSecurityLevelPortMapping, DiagnosticServiceDataIdentifierPortMapping, DiagnosticServiceGenericMapping, DiagnosticServiceSwMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.1: DiagnosticSwMapping

Of course, the specifics of the `PortPrototype` on the side of the `AUTOSAR Adaptive Diagnostic Management` need to be derived from the configuration (in this case, the definition of a `DiagnosticDataIdentifier`) of the external behavior of the diagnostic stack on the *AUTOSAR adaptive platform*, as described by a corresponding `Diagnostic Extract` [19].

A further kind of mapping that is necessary to enable diagnostics on the *AUTOSAR adaptive platform* comes with slightly more complexity.

In this case use-cases are implemented that may or may not involve several communication ends (in the form of `PortPrototype`s).

Class	**DiagnosticDataIdentifier**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents the ability to model a diagnostic data identifier (DID) that is fully specified regarding the payload at configuration-time. **Tags:**atp.recommendedPackage=DiagnosticDataIdentifiers			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticAbstractDataIdentifier*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataElement	DiagnosticParameter	*	aggr	This is the dataElement associated with the DiagnosticDataIdentifier. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataElement.bitOffset, dataElement.variationPoint.shortLabel vh.latestBindingTime=postBuild
didSize	PositiveInteger	0..1	attr	This attribute indicates the size in bytes of the DiagnosticDataIdentifier.
representsVin	Boolean	0..1	attr	This attributes indicates whether the specific DiagnosticDataIdentifier represents the vehicle identification.
supportInfoByte	DiagnosticSupportInfoByte	0..1	aggr	This attribute represents the supported information associated with the DiagnosticDataIdentifier.

Table 4.2: DiagnosticDataIdentifier

The response to this situation on the *AUTOSAR classic platform* has been the definition of the SwcServiceDependency that allows for associating several PortPrototypes in specific roles to a given use-case.

Although the need for involving different PortPrototypes in the implementation of a given use case might slightly have gone down, there is still enough motivation to keep using this pattern on the *AUTOSAR adaptive platform* as well.

For example, one benefit of this approach over a seemingly more straightforward implementation to refer to a PortPrototype directly is the ability to let several PortPrototypes (where e.g. some may represent server functionality, and the rest could represent client functionality) in concert in order to implement a given use case.

Figure 4.2 provides a visual explanation of how this kind of diagnostic mapping to model elements on the *AUTOSAR adaptive platform* works.

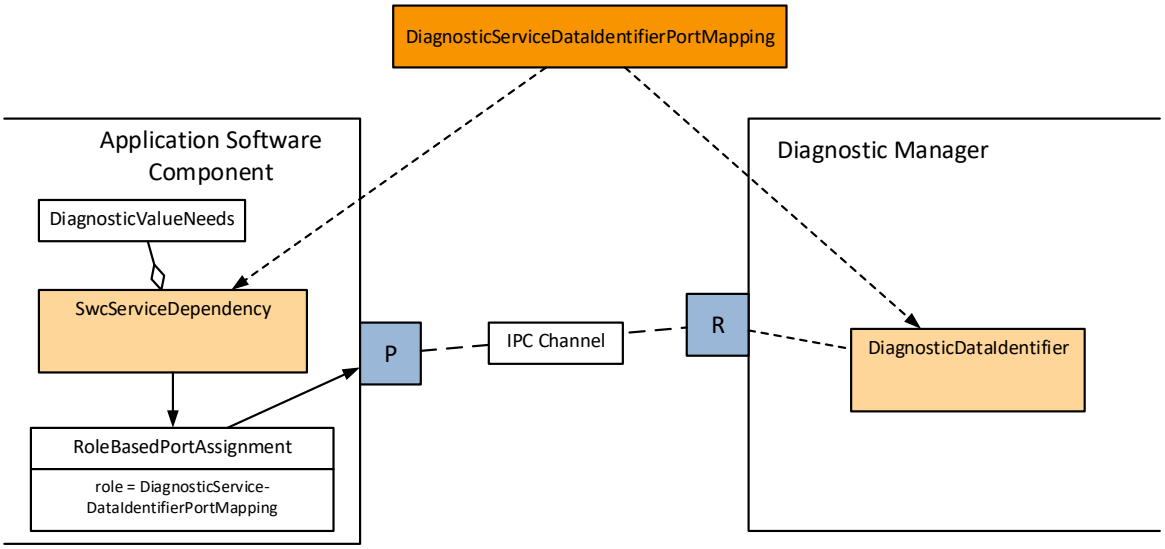


Figure 4.2: Example mapping to associate a PortPrototype with a DiagnosticDataIdentifier

Please note that the mapping targets[1] within a set of diagnostic mappings may exist in several instances at run-time.

This kind of multiple instantiation is formalized by the existence of meta-class Process (which in turn is represented by meta-class ProcessDesign on design level), see chapter 3.19.

It is very typical that different instances of a piece of application software could require a different diagnostic mapping and the modeling needs to accommodate to this requirement, i.e. a relation between a diagnostic mapping and the ProcessDesign needs to be established.

[1] on the end of the application software

As depicted by Figure 4.3, the application of a DiagnosticMapping that targets SwcServiceDependency on the *AUTOSAR adaptive platform* requires an aggregation chain from of AdaptiveApplicationSwComponentType (see section 3.2) via AdaptiveSwcInternalBehavior down to SwcServiceDependency.

[constr_10002]{DRAFT} **Only one mapping per PortPrototype** ⌈If one instance of the following sub-classes of DiagnosticSwMapping refers to a PortPrototype then no other instance of DiagnosticSwMapping shall refer to the same PortPrototype:

- DiagnosticEventPortMapping that is associated with a RPortPrototype typed by a DiagnosticMonitorInterface or a DiagnosticEventInterface.
- DiagnosticOperationCyclePortMapping that is associated with a RPortPrototype typed by a DiagnosticOperationCycleInterface.
- DiagnosticEnableConditionPortMapping that is associated with a RPortPrototype typed by a DiagnosticConditionInterface.
- DiagnosticClearConditionPortMapping that is associated with a RPortPrototype typed by a DiagnosticConditionInterface.
- DiagnosticIndicatorPortMapping that is associated with a RPortPrototype typed by a DiagnosticIndicatorInterface.
- DiagnosticMemoryDestinationPortMapping that is associated with an RPortPrototype typed by a DiagnosticDTCInformationInterface.
- DiagnosticSecurityLevelPortMapping that is associated with an PPortPrototype typed by a DiagnosticSecurityLevelInterface.
- DiagnosticServiceDataIdentifierPortMapping that is associated with a PPortPrototype typed by a DiagnosticDataIdentifierInterface, or DiagnosticDataElementInterface.

⌋()

The rationale for the existence of [constr_10002] is that the respective PortPrototype has a clearly defined functionality. For example, it can only provide the content of one DID, but it cannot provide the content of an arbitrary number of DIDs.

For such a case, the DiagnosticServiceGenericMapping (see section 4.1.11) shall be applied.

Please note the [constr_10002] does not apply to the DiagnosticServiceGenericMapping, i.e. a PortPrototype that is not subject to [constr_10002] can be referenced by multiple DiagnosticServiceGenericMapping.

In other words, the ability for several DiagnosticServiceGenericMappings to refer to the same PortPrototype is what makes the DiagnosticServiceGenericMapping *generic*.

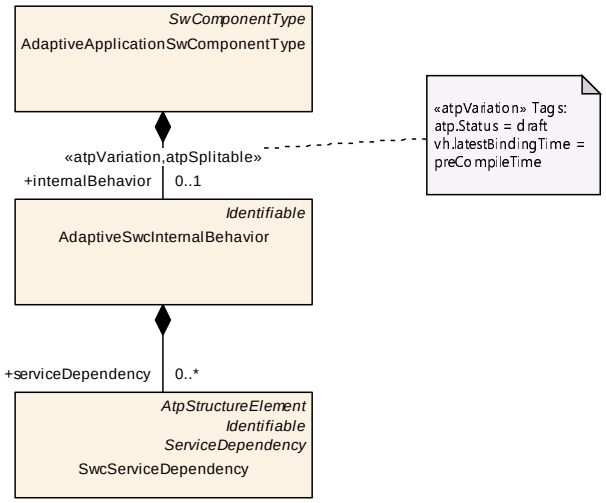


Figure 4.3: Modeling of internal behavior for the modeling of `DiagnosticMapping` that targets `SwcServiceDependency`

Class	**SwcServiceDependency**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServiceMapping			
Note	Specialization of ServiceDependency in the context of an SwcInternalBehavior. It allows to associate ports, port groups and (in special cases) data defined for an atomic software component to a given ServiceNeeds element.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceDependency*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
assignedData	RoleBasedData Assignment	*	aggr	Defines the role of an associated data object of the same component. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
assignedPort	RoleBasedPort Assignment	*	aggr	Defines the role of an associated port of the same component. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=assignedPort, assignedPort.variation Point.shortLabel vh.latestBindingTime=preCompileTime
representedPort Group	PortGroup	0..1	ref	This reference specifies an association between the ServiceNeeeds and a PortGroup, for example to request a communication mode which applies for communication via these ports. The referred PortGroup shall be local to this atomic SWC, but via the links between the Port Groups, a tool can evaluate this information such that all the ports linked via this port group on the same ECU can be found.
serviceNeeds	ServiceNeeds	0..1	aggr	The associated ServiceNeeds.

Table 4.3: SwcServiceDependency

4.1.2 Diagnostic Event to Port Mapping

[TPS_MANI_01048]{DRAFT} **Mapping of `DiagnosticEvent` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticEvent` and one or many `PortPrototypes` is created by using the `DiagnosticEventPortMapping` that refers to a `DiagnosticEvent` in the role `diagnosticEvent` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`.⌋*(RS_MANI_00005, RS_MANI_00061)*

[TPS_MANI_01336]{DRAFT} **Two use cases for using the `DiagnosticEventPortMapping`** ⌈There are two use cases for using the `DiagnosticEventPortMapping`:

- The `DiagnosticEventPortMapping` refers to a `SwcServiceDependency` that aggregates `DiagnosticEventNeeds` and that refers to an `RPortPrototype` typed by a `DiagnosticMonitorInterface` in the role `assignedPort.portPrototype`. This use case supports the reporting of diagnostic events.
- The `DiagnosticEventPortMapping` refers to a `SwcServiceDependency` that aggregates `DiagnosticEventInfoNeeds` and that refers to an `RPortPrototype` typed by a `DiagnosticEventInterface` in the role `assignedPort.portPrototype`. This use case supports the retrieval of information about a diagnostic event.

⌋*(RS_MANI_00005, RS_MANI_00061)*

The use case (as described within [TPS_MANI_01336]) that supports the reporting of a diagnostic event is depicted in Figure 4.4.

The use case (as described within [TPS_MANI_01336]) that supports the retrieval of information about a diagnostic event is depicted in Figure 4.5.

[constr_1500]{DRAFT} **Target `SwcServiceDependency` of `DiagnosticEventPortMapping.swcServiceDependencyInExecutable`** ⌈Any particular `SwcServiceDependency` that is referenced in the role `DiagnosticEventPortMapping.swcServiceDependencyInExecutable` shall

- **only** be aggregated in the role `serviceDependency` by an `AdaptiveSwcInternalBehavior`
- aggregate either a `DiagnosticEventNeeds` or a `DiagnosticEventInfoNeeds`.

⌋*()*

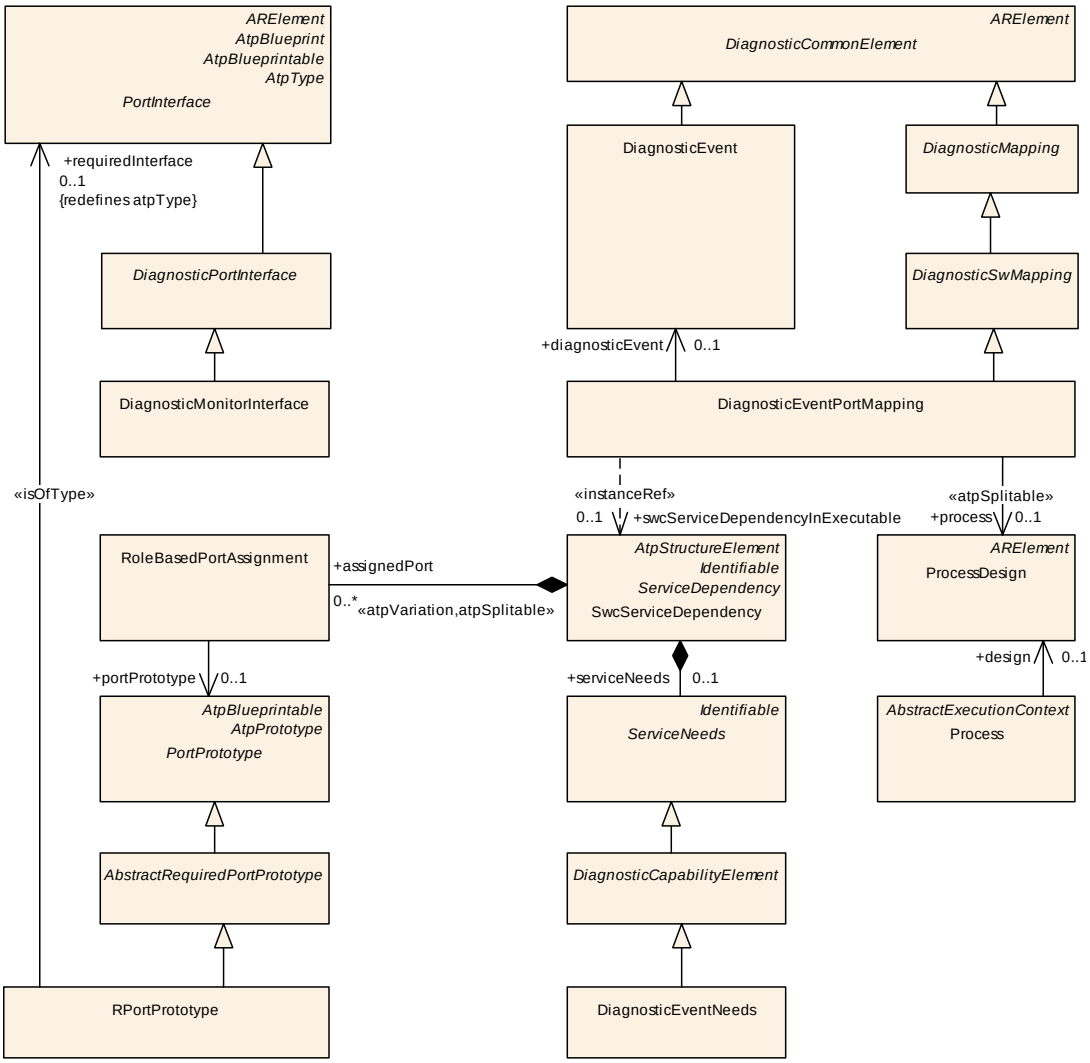


Figure 4.4: Modeling of `DiagnosticEventPortMapping` for reporting events on the *AUTOSAR adaptive platform*

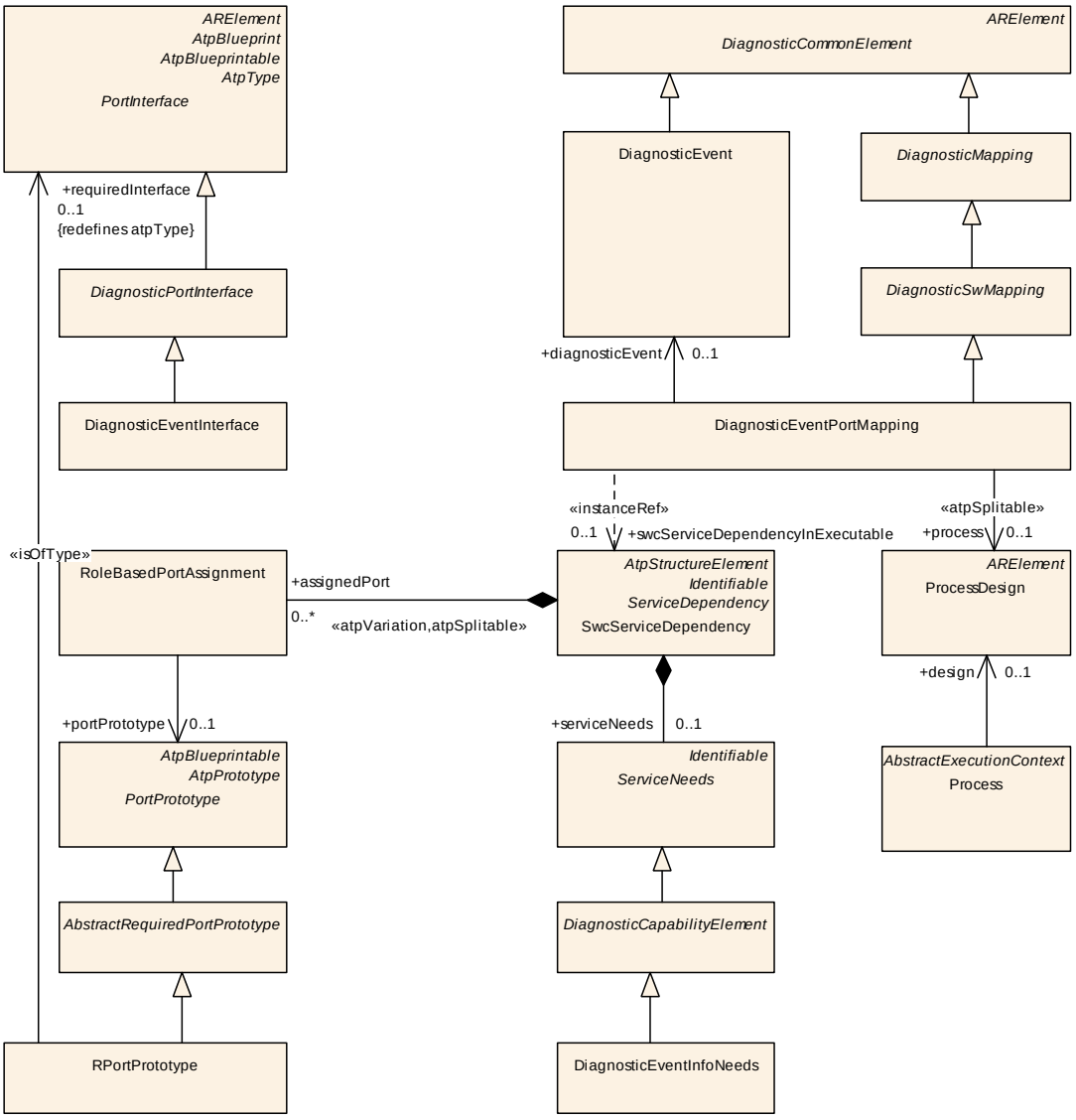


Figure 4.5: Modeling of `DiagnosticEventPortMapping` tor retrieve information about a diagnostic event on the *AUTOSAR adaptive platform*

Class	**DiagnosticEvent**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticEvent			
Note	This element is used to configure DiagnosticEvents. **Tags:**atp.recommendedPackage=DiagnosticEvents			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**DiagnosticEvent**			
associated Event Identification	PositiveInteger	0..1	attr	This attribute represents the identification number that is associated with the enclosing DiagnosticEvent and allows to identify it when placed into a snapshot record or extended data record storage. This value can be reported as internal data element in snapshot records or extended data records.
clearEvent Allowed Behavior	DiagnosticClearEvent AllowedBehaviorEnum	0..1	attr	This attribute defines the resulting UDS status byte for the related event, which shall not be cleared according to the ClearEventAllowed callback
confirmation Threshold	PositiveInteger	0..1	attr	This attribute defines the number of operation cycles with a failed result before a confirmed DTC is set to 1. The semantic of this attribute is a by "1" increased value compared to the confirmation threshold of the "trip counter" mentioned in ISO 14229-1 in figure D.4. A value of "1" defines the immediate confirmation of the DTC along with the first reported failed. This is also sometimes called "zero trip DTC". A value of "2" defines a DTC confirmation in the operation cycle after the first occurred failed. A value of "2" is typically used in the US for OBD DTC confirmation. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
connected Indicator	DiagnosticConnected Indicator	*	aggr	Event specific description of Indicators. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=connectedIndicator.shortName, connectedIndicator.variationPoint.shortLabel vh.latestBindingTime=postBuild
eventClear Allowed	DiagnosticEventClear AllowedEnum	0..1	attr	This attribute defines whether the Dem has access to a "ClearEventAllowed" callback.
prestorage FreezeFrame	Boolean	0..1	attr	This attribute describes whether the Prestorage of Freeze Frames is supported by the assigned event or not. True: Prestorage of FreezeFrames is supported False: Prestorage of FreezeFrames is not supported
prestored Freezeframe StoredInNvm	Boolean	0..1	attr	If the Event uses a prestored freeze-frame (using the operations PrestoreFreezeFrame and ClearPrestored FreezeFrame of the service interface DiagnosticMonitor) this attribute indicates if the Event requires the data to be stored in non-volatile memory. TRUE = Dem shall store the prestored data in non-volatile memory, FALSE = Data can be lost at shutdown (not stored in Nvm)
recoverableIn SameOperation Cycle	Boolean	0..1	attr	If the attribute is set to true then reporting PASSED will reset the indication of a failed test in the current operation cycle. If the attribute is set to false then reporting PASSED will be ignored and not lead to a reset of the indication of a failed test.

Table 4.4: DiagnosticEvent

Class	**DiagnosticEventPortMapping**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMapping
Note	Defines to which SWC service ports with DiagnosticEventNeeds the DiagnosticEvent is mapped. **Tags:**atp.recommendedPackage=DiagnosticMappings

▽

△

Class	**DiagnosticEventPortMapping**			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bswService Dependency	BswService DependencyIdent	0..1	ref	Reference to a BswServiceDependency that links Service Needs to BswModuleEntries.
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to the DiagnosticEvent that is assigned to SWC service ports with DiagnosticEventNeeds.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
swcFlatService Dependency	SwcService Dependency	0..1	ref	Reference to a SwcServiceDependencyType that links ServiceNeeds to SWC service ports.
swcService DependencyIn Executable	SwcService Dependency	0..1	iref	This aggregation allows for the usage of the Diagnostic EventPortMapping on the AUTOSAR adaptive platform. **Tags:**atp.Status=draft **InstanceRef implemented by:**SwcServiceDependency InExecutableInstanceRef
swcService DependencyIn System	SwcService Dependency	0..1	iref	Instance reference to a SwcServiceDependency that links ServiceNeeds to SWC service ports. **InstanceRef implemented by:**SwcServiceDependency InSystemInstanceRef

Table 4.5: DiagnosticEventPortMapping

Class	**DiagnosticEventNeeds**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs on the configuration of the Diagnostic Event Manager for one diagnostic event. Its shortName can be regarded as a symbol identifying the diagnostic event from the viewpoint of the component or module which owns this element. In case the diagnostic event specifies a production error, the shortName shall be the name of the production error.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
considerPto Status	Boolean	0..1	attr	PTO (Power Take Off) has an impact on the respective emission-related event (OBD). This information shall be provided by SW-C description in order to consider the PTO relevance e.g. for readiness (PID $01) computation. For events with dtcKind set to 'nonEmmissionRelatedDtc' this attribute is typically false.
deferringFid	FunctionInhibitionNeeds	*	ref	This reference contains the link to a function identifier within the FiM which is used by the monitor before delivering a result.
diagEvent Debounce Algorithm	DiagEventDebounce Algorithm	0..1	aggr	Specifies the abstract need on the Debounce Algorithm applied by the Diagnostic Event Manager.

▽

△

Class	**DiagnosticEventNeeds**			
dtcKind	DtcKindEnum	0..1	attr	This attribute indicates the kind of the diagnostic monitor according to the SWS Diagnostic Event Manger. This attribute applies for the UDS diagnostics use case.
obdDtcNumber	PositiveInteger	0..1	attr	This represents a reasonable Diagnostic Trouble Code. This allows to predefine the Diagnostic Trouble Code, e.g. if the a function developer has received a particular requirement from the OEM or from a standardization body. This attribute applies for the OBD diagnostics use case.
prestored Freezeframe StoredInNvm	Boolean	0..1	attr	If the Event uses a prestored freeze-frame (using the operations PrestoreFreezeFrame and ClearPrestored FreezeFrame of the service interface DiagnosticMonitor) this attribute indicates if the Event requires the data to be stored in non-volatile memory. TRUE = Dem shall store the prestored data in non-volatile memory, FALSE = Data can be lost at shutdown (not stored in Nvm).
reportBehavior	ReportBehaviorEnum	0..1	attr	This switch indicates whether or not the BSW module is allowed to report the related Events before Dem_Init().
udsDtcNumber	PositiveInteger	0..1	attr	This represents a reasonable Diagnostic Trouble Code. This allows to predefine the Diagnostic Trouble Code, e.g. if the a function developer has received a particular requirement from the OEM or from a standardization body. This attribute applies for the UDS diagnostics use case.
usesMonitor Data	Boolean	0..1	attr	This attribute defines whether additional monitor data shall be added to the reporting of events.

Table 4.6: DiagnosticEventNeeds

Class	**DiagnosticEventInfoNeeds**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the needs of a software-component interested to get information regarding specific DTCs.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dtcKind	DtcKindEnum	0..1	attr	This attribute indicates the kind of the diagnostic event according to the SWS Diagnostic Event Manger for which the DiagnosticInfo is requested. This attribute applies for the UDS diagnostics use case.
obdDtcNumber	PositiveInteger	0..1	attr	This represents a reasonable Diagnostic Trouble Code. This allows to predefine the Diagnostic Trouble Code, e.g. if the function developer has received a particular requirement from the OEM or from a standardization body. This attribute applies for the OBD diagnostics use case.

▽

△

Class	**DiagnosticEventInfoNeeds**			
udsDtcNumber	PositiveInteger	0..1	attr	This represents a reasonable Diagnostic Trouble Code. This allows to predefine the Diagnostic Trouble Code, e.g. if the function developer has received a particular requirement from the OEM or from a standardization body. This attribute applies for the UDS diagnostics use case.

Table 4.7: DiagnosticEventInfoNeeds

4.1.3 Diagnostic Operation Cycle to Port Mapping

[TPS_MANI_01049]{DRAFT} **Mapping of `DiagnosticOperationCycle` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticOperationCycle` and one or many `PortPrototype`s is created by using the `DiagnosticOperationCyclePortMapping` that refers to a `DiagnosticOperationCycle` in the role `operationCycle` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`.⌋*(RS_MANI_00005, RS_MANI_00061)*

[constr_1501]{DRAFT} **Target `SwcServiceDependency` of `DiagnosticOperationCyclePortMapping.swcServiceDependencyInExecutable`** ⌈Any particular `SwcServiceDependency` that is referenced in the role `DiagnosticOperationCyclePortMapping.swcServiceDependencyInExecutable` shall

- **only** be aggregated in the role `serviceDependency` by an `AdaptiveSwcInternalBehavior` and
- aggregate a `DiagnosticOperationCycleNeeds`.

⌋*()*

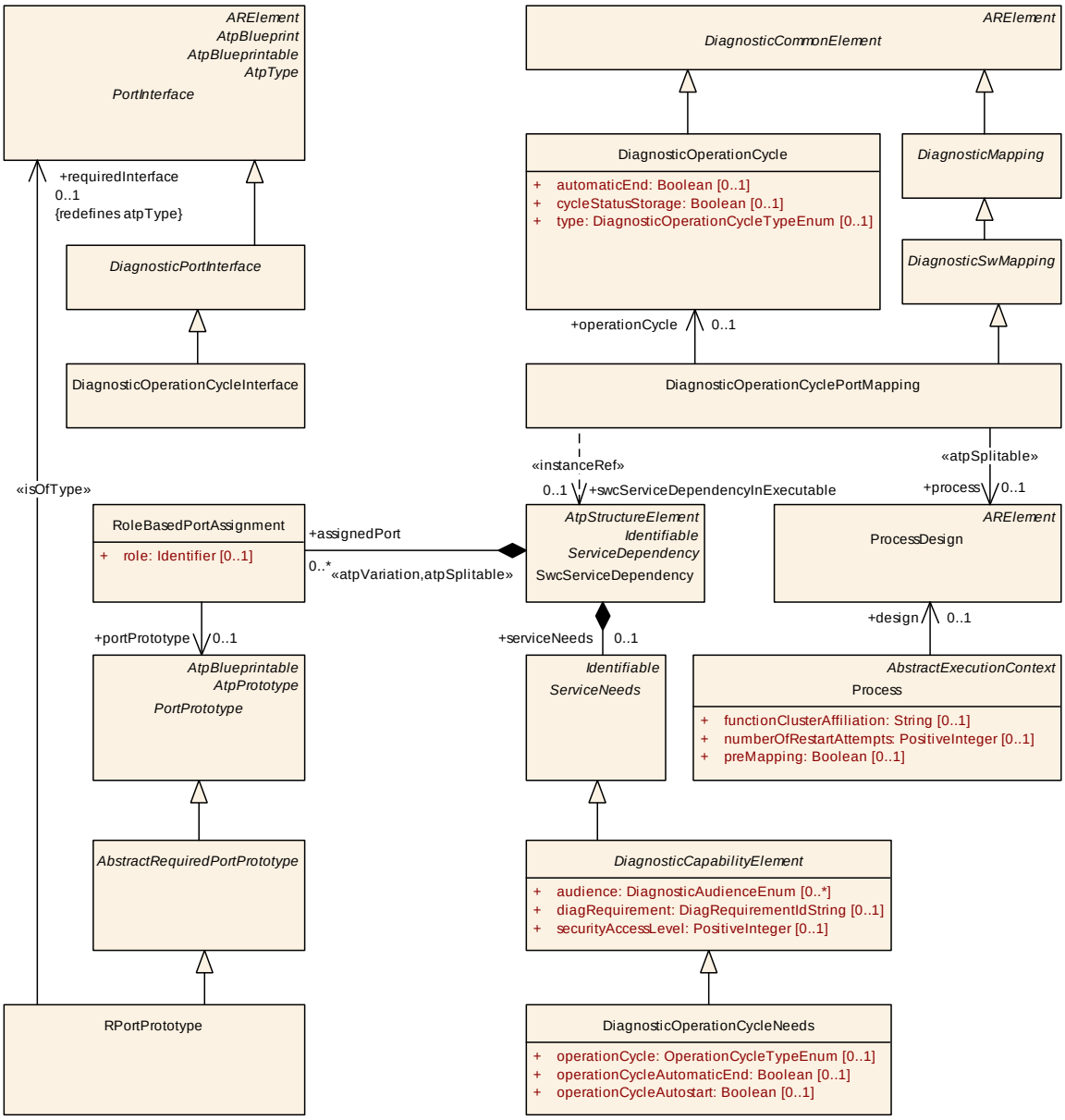


Figure 4.6: Modeling of `DiagnosticOperationCyclePortMapping` for the usage on the *AUTOSAR adaptive platform*

Class	**DiagnosticOperationCycle**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticOperationCycle			
Note	Definition of an operation cycle that is the base of the event qualifying and for Dem scheduling. **Tags:**atp.recommendedPackage=DiagnosticOperationCycles			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**DiagnosticOperationCycle**			
automaticEnd	Boolean	0..1	attr	If set to true the driving cycle shall automatically end at either Dem_Shutdown() or Dem_Init(). This attribute is only relevant for the AUTOSAR adaptive platform. It no longer has a meaning on the AUTOSAR classic platform.
cycleStatus Storage	Boolean	0..1	attr	Defines if the operation cycle state is available over the power cycle (stored non-volatile) or not. • true: the operation cycle state is stored non-volatile • false: the operation cycle state is only stored volatile This attribute is only relevant for the AUTOSAR adaptive platform. It no longer has a meaning on the AUTOSAR classic platform.
type	DiagnosticOperation CycleTypeEnum	0..1	attr	Operation cycles types for the Dem.

Table 4.8: DiagnosticOperationCycle

Class	**DiagnosticOperationCyclePortMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticOperationCycleNeeds the DiagnosticOperationCycle is mapped. **Tags:**atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
operationCycle	DiagnosticOperation Cycle	0..1	ref	Reference to the DiagnosticOperationCycle that is assigned to SWC service ports with DiagnosticOperation CycleNeeds.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
swcFlatService Dependency	SwcService Dependency	0..1	ref	Reference to a SwcServiceDependencyType that links ServiceNeeds to SWC service ports.
swcService DependencyIn Executable	SwcService Dependency	0..1	iref	This aggregation allows for the usage of the Diagnostic OperationCyclePortMapping on the AUTOSAR adaptive platform. **Tags:**atp.Status=draft **InstanceRef implemented by:**SwcServiceDependency InExecutableInstanceRef
swcService DependencyIn System	SwcService Dependency	0..1	iref	Instance reference to a SwcServiceDependency that links ServiceNeeds to SWC service ports. **InstanceRef implemented by:**SwcServiceDependency InSystemInstanceRef

Table 4.9: DiagnosticOperationCyclePortMapping

Class	**DiagnosticOperationCycleNeeds**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the needs of a software-component to provide information regarding the operation cycle management to the Dem module.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
operationCycle	OperationCycleType Enum	0..1	attr	Operation cycles types for the Dem to be supported by cycle-state APIs.
operationCycle AutomaticEnd	Boolean	0..1	attr	If this attribute is set to true the Dem shall automatically end the driving cycle at either Dem_Shutdown() or Dem_ Init().
operationCycle Autostart	Boolean	0..1	attr	If this attribute is set to true the operation cycles is automatically (re-)started during Dem_PreInit().

Table 4.10: DiagnosticOperationCycleNeeds

4.1.4 Diagnostic Enable Condition to Port Mapping

[TPS_MANI_01050]{DRAFT} **Mapping of `DiagnosticEnableCondition` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticEnableCondition` and one or many `PortPrototype`s is created by using the `DiagnosticEnableConditionPortMapping` that refers to a `DiagnosticEnableCondition` in the role `enableCondition` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`.⌋*(RS_MANI_00005, RS_MANI_00061)*

[constr_1502]{DRAFT} **Target `SwcServiceDependency` of `DiagnosticEnableConditionPortMapping.swcServiceDependencyInExecutable`** ⌈Any particular `SwcServiceDependency` that is referenced in the role `DiagnosticEnableConditionPortMapping.swcServiceDependencyInExecutable` shall

- **only** be aggregated in the role `serviceDependency` by an `AdaptiveSwcInternalBehavior`
- aggregate a `DiagnosticEnableConditionNeeds`.

⌋*()*

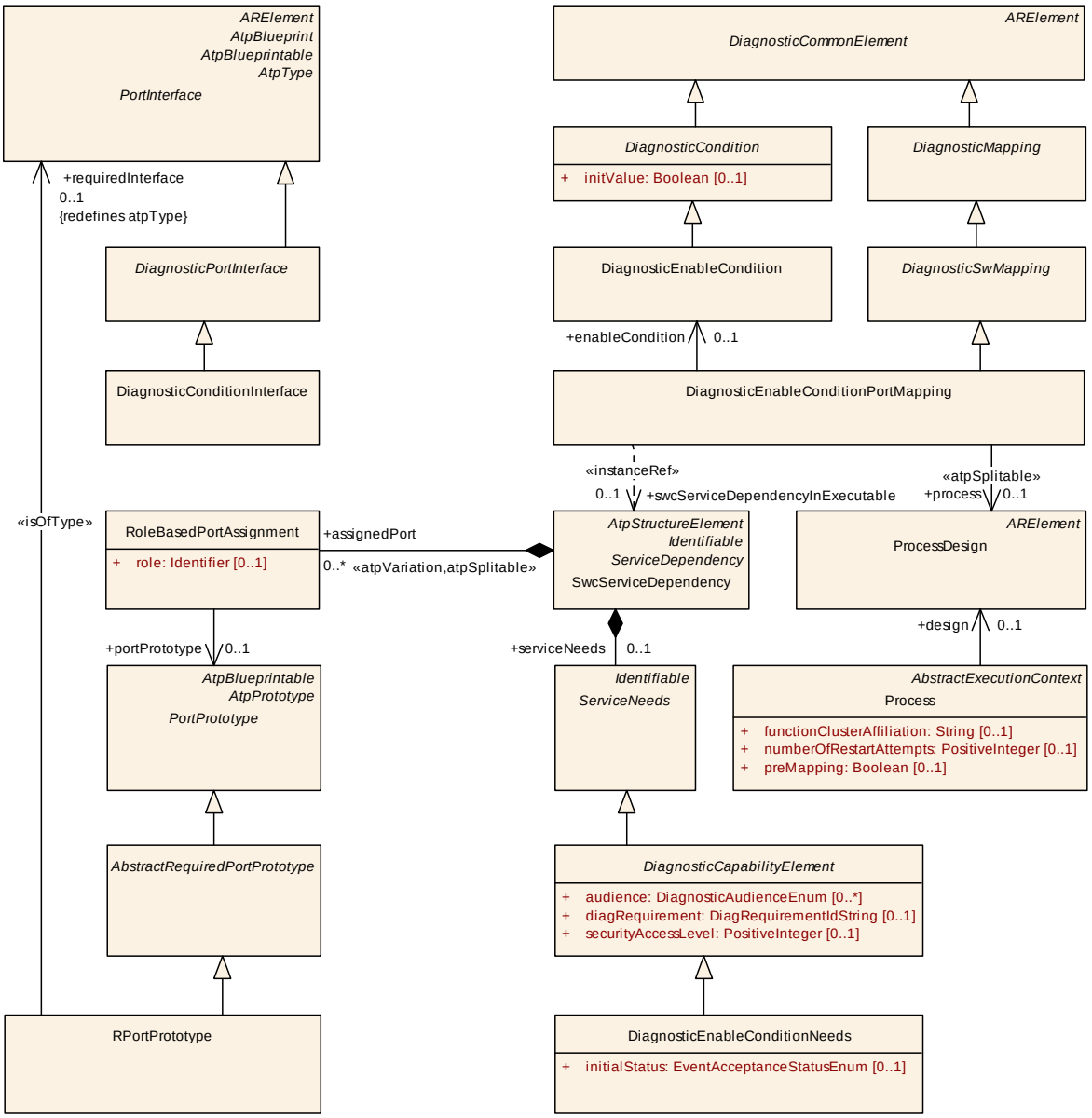


Figure 4.7: Modeling of `DiagnosticEnableConditionPortMapping` for the usage on the *AUTOSAR adaptive platform*

Class	**DiagnosticEnableCondition**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticCondition			
Note	Specification of an enable condition. **Tags:**atp.recommendedPackage=DiagnosticConditions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticCondition*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.11: DiagnosticEnableCondition

Class	**DiagnosticEnableConditionPortMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticEnableConditionNeeds the DiagnosticEnable Condition is mapped. **Tags:**atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
enableCondition	DiagnosticEnable Condition	0..1	ref	Reference to the EnableCondition which is mapped to a SWC service port with DiagnosticEnableConditionNeeds.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
swcFlatService Dependency	SwcService Dependency	0..1	ref	Reference to a SwcServiceDependencyType that links ServiceNeeds to SWC service ports. This reference can be used in early stages of the development in order to identify the SwcServiceDependency without a full System Context.
swcService DependencyIn Executable	SwcService Dependency	0..1	iref	This aggregation allows for the usage of the Diagnostic EnableConditionPortMapping on the AUTOSAR adaptive platform. **Tags:**atp.Status=draft **InstanceRef implemented by:**SwcServiceDependency InExecutableInstanceRef
swcService DependencyIn System	SwcService Dependency	0..1	iref	Instance reference to a SwcServiceDependency that links ServiceNeeds to SWC service ports. **InstanceRef implemented by:**SwcServiceDependency InSystemInstanceRef

Table 4.12: DiagnosticEnableConditionPortMapping

Class	**DiagnosticEnableConditionNeeds**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the needs of a software-component to provide the capability to set an enable condition.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initialStatus	EventAcceptanceStatus Enum	0..1	attr	Defines the initial status for enable or disable of acceptance of event reports of a diagnostic event.

Table 4.13: DiagnosticEnableConditionNeeds

4.1.5 Diagnostic Clear Condition to Port Mapping

[TPS_MANI_01259]{DRAFT} **Mapping of `DiagnosticClearCondition` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticClearCondition` and one or many

PortPrototypes is created by using the DiagnosticClearConditionPortMapping that refers to a DiagnosticClearCondition in the role clearCondition as well as to a SwcServiceDependency in the role swcServiceDependencyInExecutable.⌋(*RS_MANI_00005, RS_MANI_00061*)

[constr_1698]{DRAFT} **Target SwcServiceDependency of DiagnosticClearConditionPortMapping.swcServiceDependencyInExecutable** ⌈Any particular SwcServiceDependency that is referenced in the role DiagnosticClearConditionPortMapping.swcServiceDependencyInExecutable shall

- **only** be aggregated in the role serviceDependency by an AdaptiveSwcInternalBehavior and
- aggregate a DiagnosticClearConditionNeeds.

⌋()

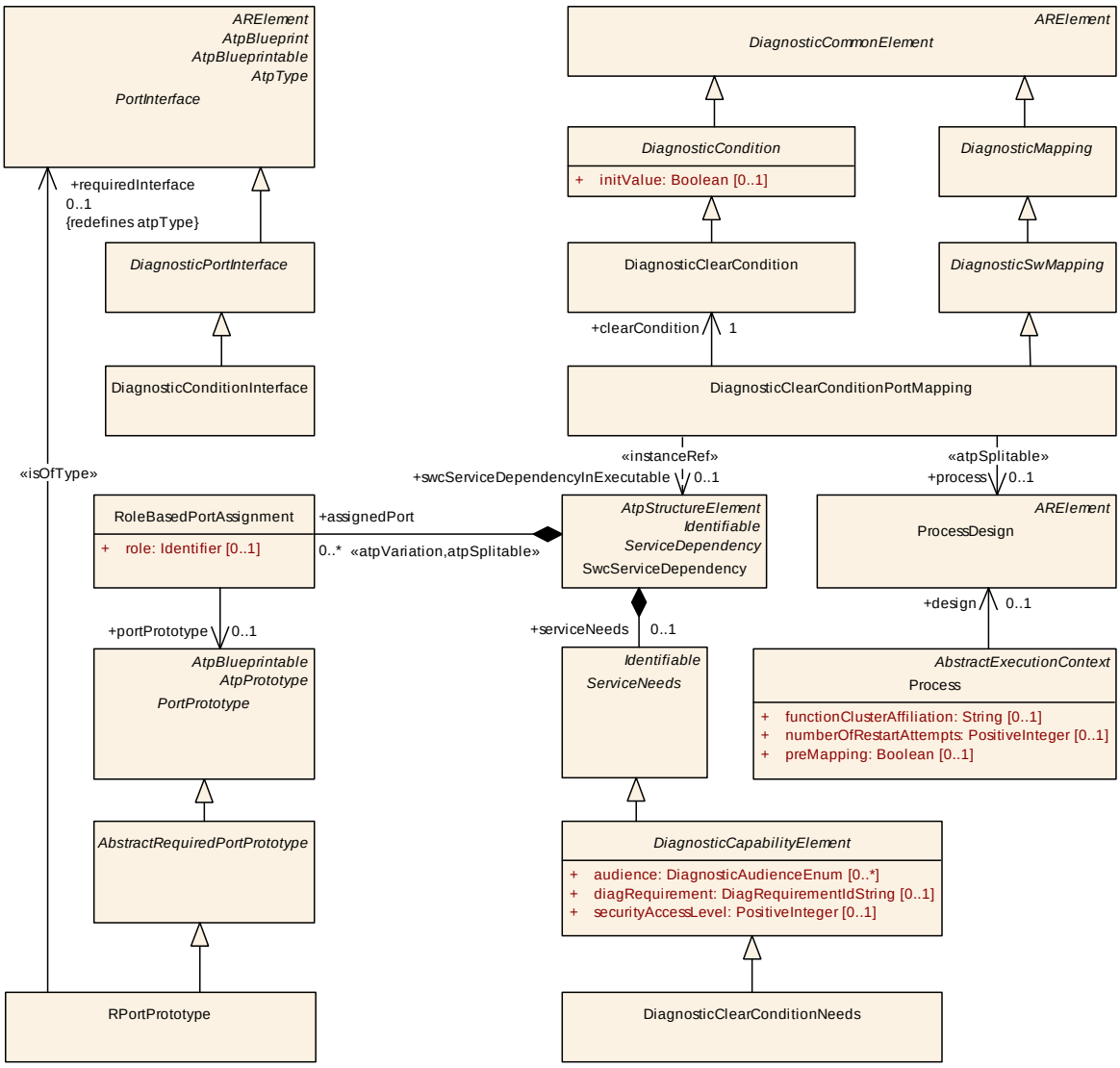


Figure 4.8: Modeling of DiagnosticClearConditionPortMapping for the usage on the *AUTOSAR adaptive platform*

Class	**DiagnosticClearConditionPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticsClearConditionNeeds the DiagnosticClearCondition is mapped. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clearCondition	DiagnosticClear Condition	1	ref	Reference to the ClearCondition which is mapped to a SWC service port with DiagnosticClearConditionNeeds. **Tags:**atp.Status=draft
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
swcService DependencyIn Executable	SwcService Dependency	0..1	iref	This aggregation allows for the usage of the Diagnostic ClearConditionPortMapping on the AUTOSAR adaptive platform. **Tags:**atp.Status=draft **InstanceRef implemented by:**SwcServiceDependency InExecutableInstanceRef

Table 4.14: DiagnosticClearConditionPortMapping

4.1.6 Diagnostic Indicator to Port Mapping

[TPS_MANI_01260]{DRAFT} **Mapping of `DiagnosticIndicator` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticIndicator` and one or many `PortPrototype`s is created by using the `DiagnosticIndicatorPortMapping` that refers to a `DiagnosticIndicator` in the role `indicator` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`.⌋*(RS_MANI_00005, RS_MANI_00061)*

[constr_1699]{DRAFT} **Target `SwcServiceDependency` of `DiagnosticIndicatorPortMapping.swcServiceDependencyInExecutable`** ⌈Any particular `SwcServiceDependency` that is referenced in the role `DiagnosticIndicatorPortMapping.swcServiceDependencyInExecutable` shall

- **only** be aggregated in the role `serviceDependency` by an `AdaptiveSwcInternalBehavior` and
- aggregate a `DiagnosticIndicatorNeeds`.

⌋*()*

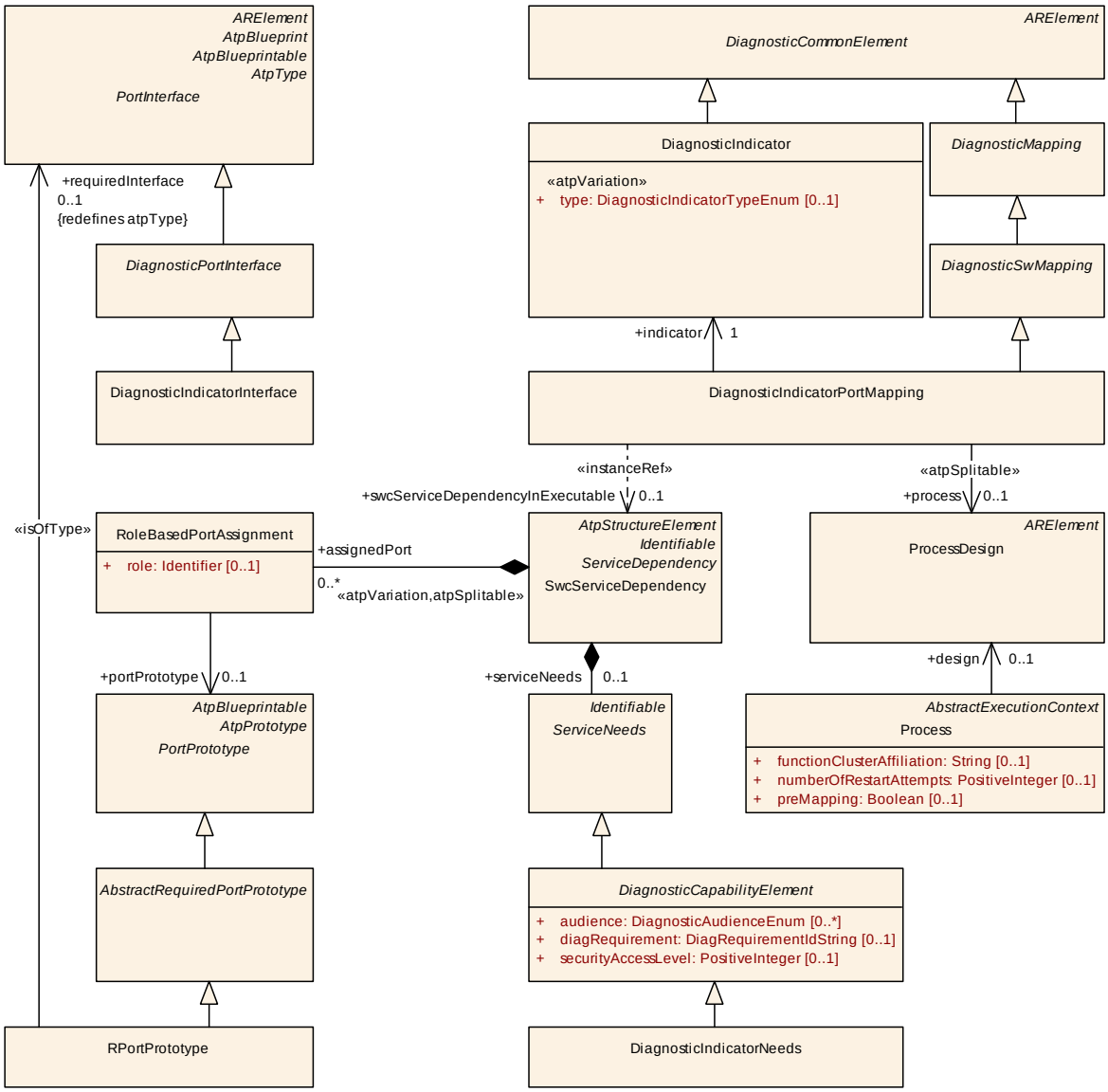


Figure 4.9: Modeling of `DiagnosticIndicatorPortMapping` for the usage on the *AUTOSAR adaptive platform*

Class	**DiagnosticIndicatorPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticsIndicatorNeeds the DiagnosticIndicator is mapped. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	DiagnosticIndicatorPortMapping			
indicator	DiagnosticIndicator	1	ref	Reference to the DiagnosticIndicator which is mapped to a SWC service port with DiagnosticIndicatorNeeds. **Tags:**atp.Status=draft
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
swcService DependencyIn Executable	SwcService Dependency	0..1	iref	This aggregation allows for the usage of the Diagnostic IndicatorPortMapping on the AUTOSAR adaptive platform. **Tags:**atp.Status=draft **InstanceRef implemented by:**SwcServiceDependency InExecutableInstanceRef

Table 4.15: DiagnosticIndicatorPortMapping

4.1.7 Diagnostic Memory Destination to Port Mapping

[TPS_MANI_01261]{DRAFT} **Mapping of `DiagnosticMemoryDestination` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticMemoryDestination` and one or many `PortPrototype`s is created by using the `DiagnosticMemoryDestinationPortMapping` that refers to a `DiagnosticMemoryDestination` in the role `memoryDestination` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`.⌋*(RS_MANI_00005, RS_MANI_00061)*

[constr_1700]{DRAFT} **Target `SwcServiceDependency` of `DiagnosticMemoryDestinationPortMapping.swcServiceDependencyInExecutable`** ⌈Any particular `SwcServiceDependency` that is referenced in the role `DiagnosticMemoryDestinationPortMapping.swcServiceDependencyInExecutable` shall

- **only** be aggregated in the role `serviceDependency` by an `AdaptiveSwcInternalBehavior` and
- aggregate a `DiagnosticEventInfoNeeds`.

⌋*()*

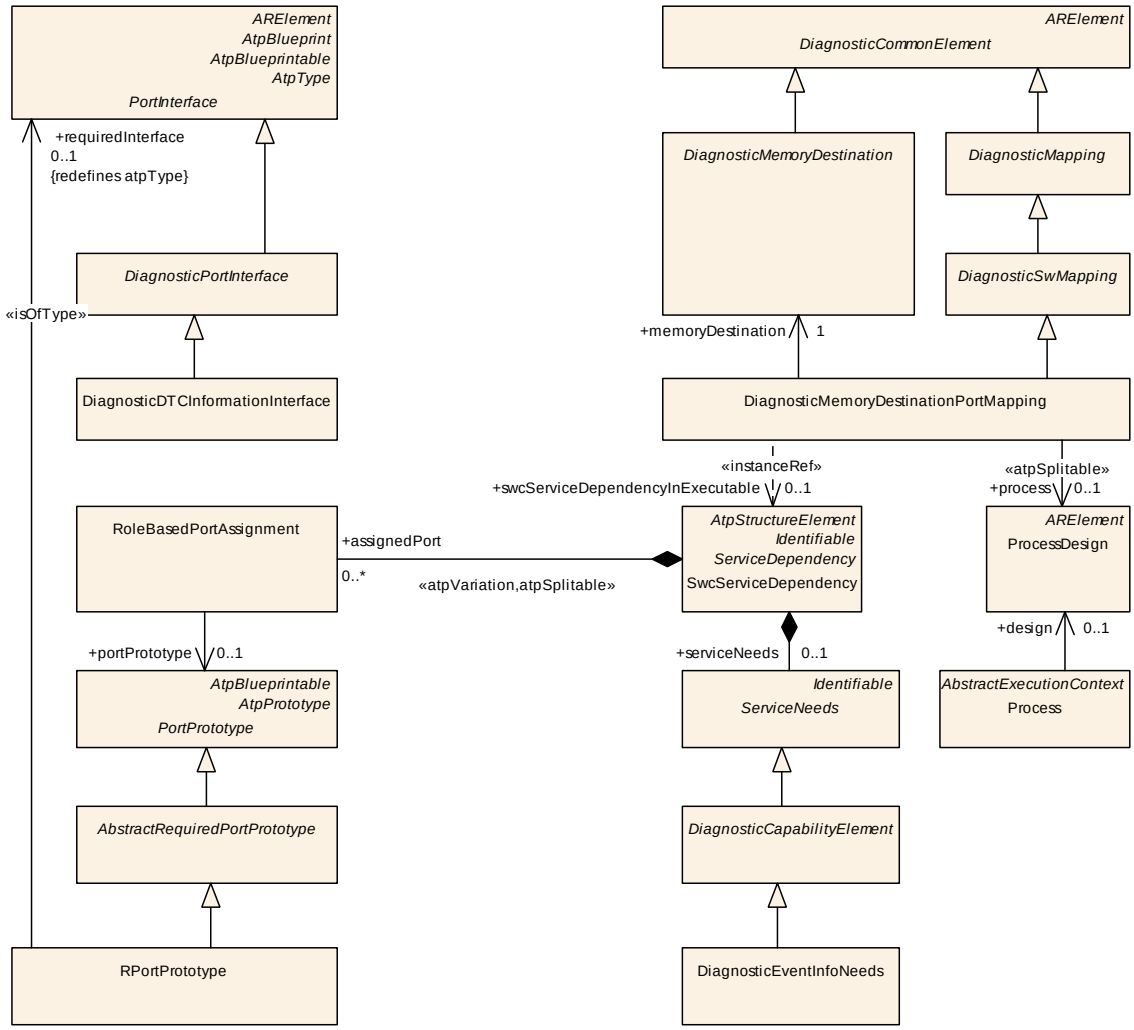


Figure 4.10: Modeling of `DiagnosticMemoryDestinationPortMapping` for the usage on the *AUTOSAR adaptive platform*

Class	**DiagnosticMemoryDestinationPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticsEventInfoNeeds the DiagnosticMemoryDestination is mapped. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
memory Destination	DiagnosticMemory Destination	1	ref	Reference to the MemoryDestination which is mapped to a SWC service port with DiagnosticEventInfoNeeds. **Tags:**atp.Status=draft

△

Class	**DiagnosticMemoryDestinationPortMapping**			
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
swcService DependencyIn Executable	SwcService Dependency	0..1	iref	This aggregation allows for the usage of the Diagnostic MemoryDestinationMapping on the AUTOSAR adaptive platform. **Tags:**atp.Status=draft **InstanceRef implemented by:**SwcServiceDependency InExecutableInstanceRef

Table 4.16: DiagnosticMemoryDestinationPortMapping

4.1.8 Diagnostic Security to Port Mapping

[TPS_MANI_01262]{DRAFT} **Mapping of `DiagnosticSecurityLevel` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticSecurityLevel` and one or many `PortPrototype`s is created by using the `DiagnosticSecurityLevelPortMapping` that refers to a `DiagnosticSecurityLevel` in the role `securityLevel` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`.⌋*(RS_MANI_00005, RS_MANI_00061)*

[constr_1701]{DRAFT} **Target `SwcServiceDependency` of `DiagnosticSecurityLevelPortMapping.swcServiceDependencyInExecutable`** ⌈Any particular `SwcServiceDependency` that is referenced in the role `DiagnosticSecurityLevelPortMapping.swcServiceDependencyInExecutable` shall

- **only** be aggregated in the role `serviceDependency` by an `AdaptiveSwcInternalBehavior` and
- aggregate a `DiagnosticsCommunicationSecurityNeeds`.

⌋*()*

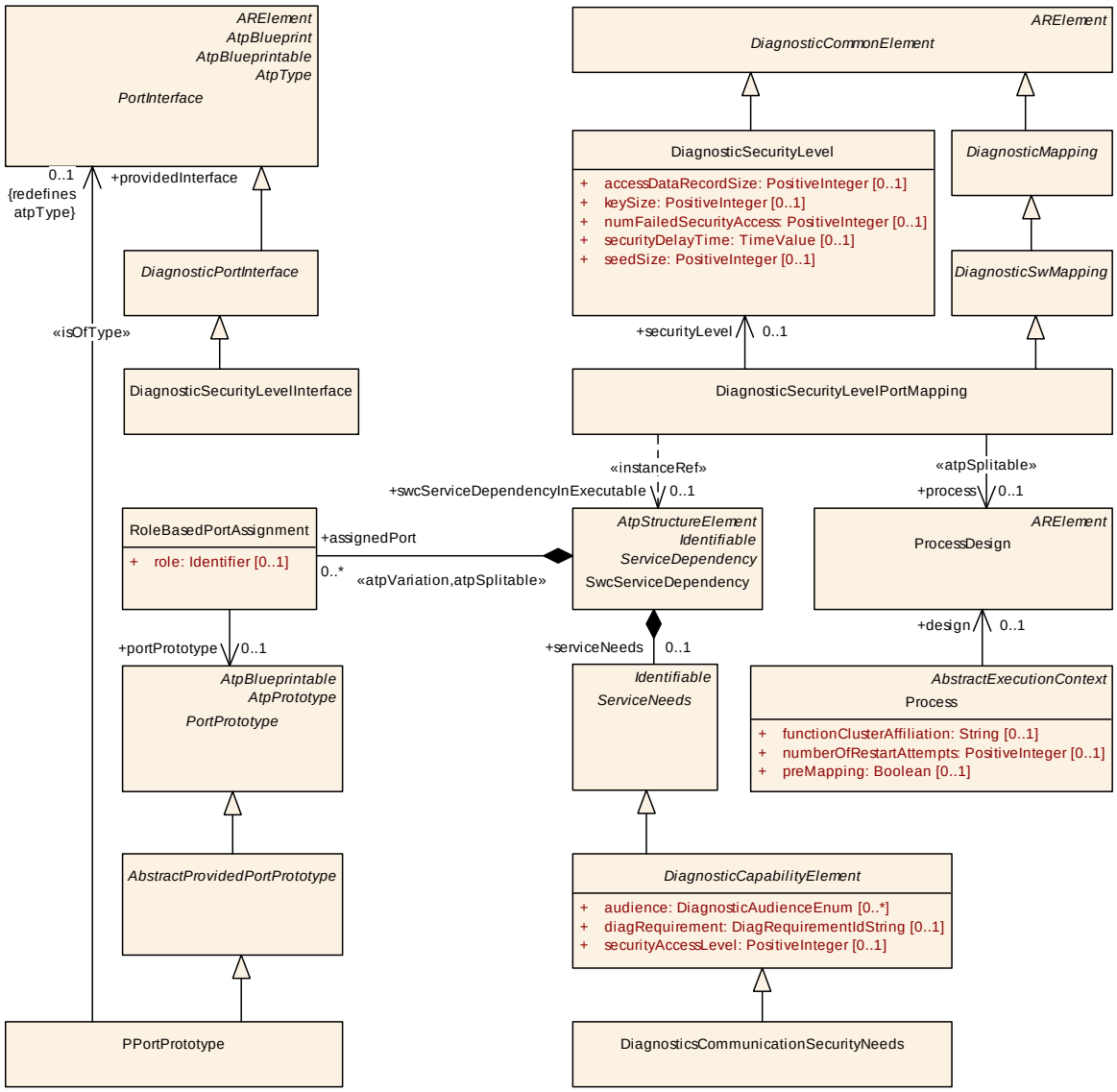


Figure 4.11: Modeling of `DiagnosticSecurityLevelPortMapping` for the usage on the *AUTOSAR adaptive platform*

Class	**DiagnosticSecurityLevelPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticsCommunicationSecurityNeeds the Diagnostic SecurityLevel is mapped. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**DiagnosticSecurityLevelPortMapping**			
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
securityLevel	DiagnosticSecurityLevel	0..1	ref	Reference to the SecurityLevelwhich is mapped to a SWC service port with DiagnosticCommunicationSecurity Needs. **Tags:**atp.Status=draft
swcService DependencyIn Executable	SwcService Dependency	0..1	iref	This aggregation allows for the usage of the Diagnostic SecurityLevelMapping on the AUTOSAR adaptive platform. **Tags:**atp.Status=draft **InstanceRef implemented by:**SwcServiceDependency InExecutableInstanceRef

Table 4.17: DiagnosticSecurityLevelPortMapping

Class	**DiagnosticsCommunicationSecurityNeeds**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the needs of a software-component to verify the access to security level via diagnostic services.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.18: DiagnosticsCommunicationSecurityNeeds

4.1.9 Diagnostic Data Identifier to Port Mapping

The DM on the *AUTOSAR adaptive platform* has the ability to access entire `DiagnosticDataIdentifier`s at once. For supporting this ability, a dedicated mapping class named `DiagnosticServiceDataIdentifierPortMapping` is introduced.

[TPS_MANI_01263]{DRAFT} **Mapping of `DiagnosticDataIdentifier` or `DiagnosticDataElement` to `PortPrototype`(s) on the *AUTOSAR adaptive platform***
⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticDataIdentifier` resp. `DiagnosticDataElement` and one or many `PortPrototype`s is created by using the `DiagnosticServiceDataIdentifierPortMapping` that refers to either

- a `DiagnosticDataIdentifier` in the role `diagnosticDataIdentifier` or
- a `DiagnosticDataElement` in the role `diagnosticDataElement`

as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`.⌋*(RS_MANI_00005, RS_MANI_00061)*

As depicted in Figure 4.12, DiagnosticServiceDataIdentifierPortMapping has the ability to handle access to **either** an entire DID **or** to just an element of a DID.

Therefore, the existence of [constr_10003] is required to enforce that just one of the references is actually used for any given DiagnosticServiceDataIdentifier-PortMapping.

[constr_10003]{DRAFT} **Restriction for the existence of DiagnosticServiceDataIdentifierPortMapping.diagnosticDataIdentifier vs. DiagnosticServiceDataIdentifierPortMapping.diagnosticDataElement** ⌈For each DiagnosticServiceDataIdentifierPortMapping, **either** the reference in the role diagnosticDataIdentifier **or** diagnosticDataElement shall exist.⌋ *()*

[constr_1702]{DRAFT} **Target SwcServiceDependency of DiagnosticServiceDataIdentifierPortMapping.swcServiceDependencyInExecutable** ⌈Any particular SwcServiceDependency that is referenced in the role DiagnosticServiceDataIdentifierPortMapping.swcServiceDependencyInExecutable shall

- **only** be aggregated in the role serviceDependency by an AdaptiveSwcInternalBehavior and
- aggregate a DiagnosticValueNeeds.

⌋*()*

Class	**DiagnosticServiceDataIdentifierPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	This meta-class provides the ability to define a diagnostic access to an entire DID. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticServiceMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnosticDataElement	DiagnosticDataElement	0..1	ref	This reference represents the applicable DiagnosticDataElement. **Tags:**atp.Status=draft
diagnosticDataIdentifier	DiagnosticDataIdentifier	0..1	ref	This reference represents the applicable DiagnosticDataIdentfiier. **Tags:**atp.Status=draft
process	ProcessDesign	1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft

▽

△

Class	DiagnosticServiceDataIdentifierPortMapping			
swcService DependencyIn Executable	SwcService Dependency	0..1	iref	This reference identifies the applicable SwcService Dependency. The reference has the ability to point into the component hierarchy (under possible consideration of the rootSoftwareComposition). **Tags:**atp.Status=draft **InstanceRef implemented by:**SwcServiceDependency InExecutableInstanceRef

Table 4.19: DiagnosticServiceDataIdentifierPortMapping

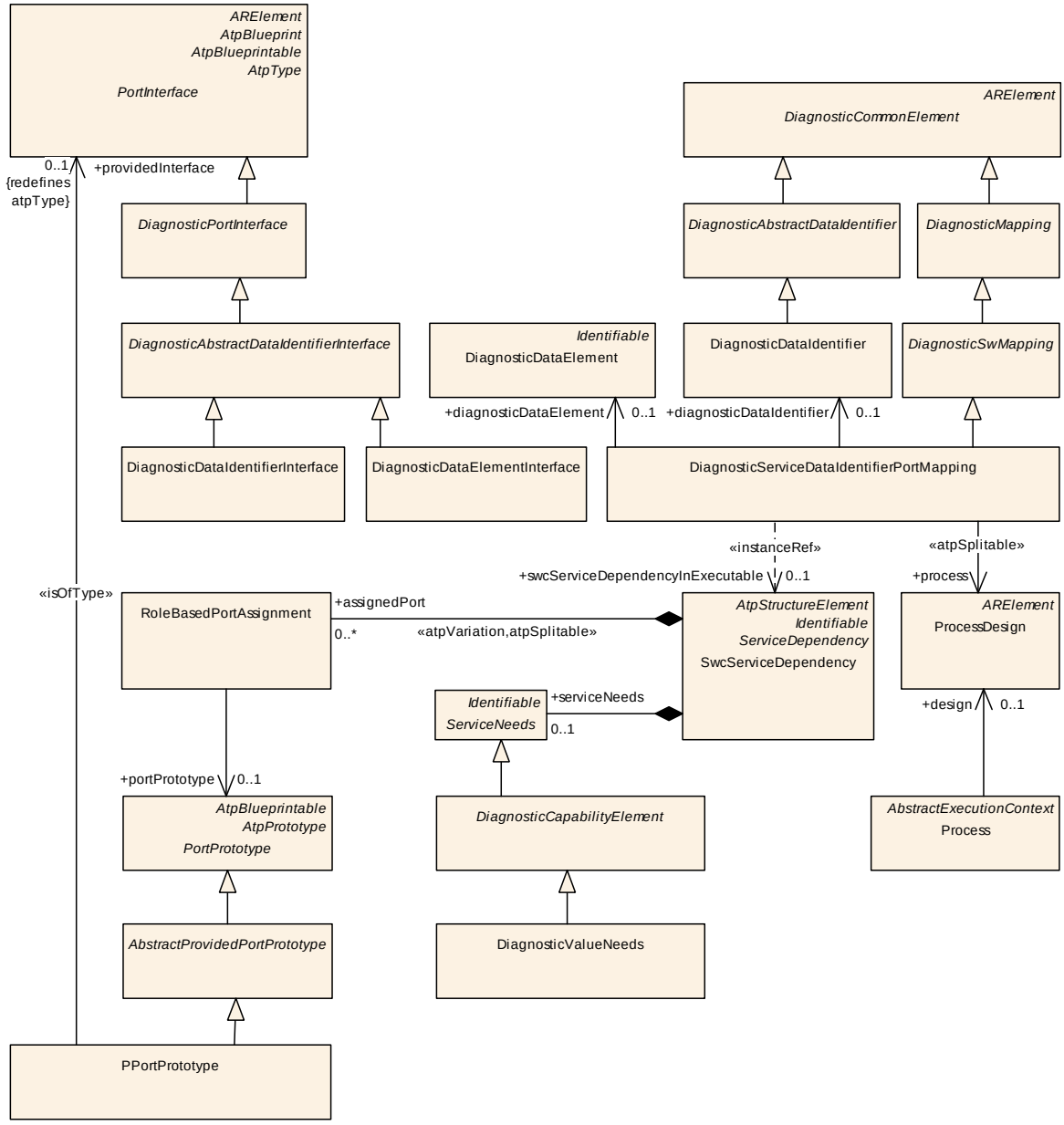


Figure 4.12: Modeling of `DiagnosticServiceDataIdentifierPortMapping` for the usage on the *AUTOSAR adaptive platform*

Class	**DiagnosticValueNeeds**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the general needs on the configuration of the Diagnostic Communication Manager (DCM) which are not related to a particular item (e.g. a PID). The main use case is the mapping of service ports to the DCM which are not related to a particular item. In the case of using a sender receiver communicated value, the related value shall be taken via assigned Data in the role "signalBasedDiagnostics". In case of using a client/server communicated value, the related value shall be communicated via the port referenced by asssignedPort. The details of this communication (e.g. appropriate naming conventions) are specified in the related software specifications (SWS).			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataLength	PositiveInteger	0..1	attr	This attribute is applicable only if the ServiceNeed is aggregated within BswModuleDependency. This attribute represents the length of data (in bytes) provided for this particular PID signal.
diagnosticValue Access	DiagnosticValueAccess Enum	0..1	attr	This attribute controls whether the data can be read and written or whether it is to be handled read-only.
didNumber	PositiveInteger	0..1	attr	This represents a Data identifier for the diagnostic value. This allows to predefine the DID number if the responsible function developer has received a particular requirement from the OEM or from a standardization body.
fixedLength	Boolean	0..1	attr	This attribute controls whether the data length of the data is fixed.
processingStyle	DiagnosticProcessing StyleEnum	0..1	attr	This attribute controls whether interaction requires the software-component to react synchronously on a request or whether it processes the request in background but still the DCM has to issue the call again to eventually obtain the result of the request.

Table 4.20: DiagnosticValueNeeds

4.1.10 Diagnostic Generic UDS Service Handler to Port Mapping

it is possible to associate a collection of UDS services to a given `PPortPrototype` with the intention that the `PPortPrototype` can handle the associated services.

By creating a dedicated association between generic UDs handlers and the services they can take it is possible to use multiple generic UDS handlers and let each take only the associated services.

Technically, a possible alternative to the documented modeling of generic UDS handling would be to avoid the mapping at all and foresee the existence of a catch-all generic UDS handler.

This, to a large extent, contradicts the idea of having modular software installations on the basis of the definition of `SoftwareCluster`s (see section 14.2).

4.1.11 Diagnostic Generic Mapping

[TPS_MANI_01326]{DRAFT} **Generic Mapping to a DiagnosticServiceInstance on the AUTOSAR Adaptive Platform** ⌈On the AUTOSAR adaptive platform, the relation between a DiagnosticServiceInstance and one or many PortPrototypes for for select diagnostic services is created by using the meta-class DiagnosticServiceGenericMapping that refers to a DiagnosticServiceInstance in the role diagnosticServiceInstance, a SwcServiceDependency in the role swcServiceDependencyInExecutable, and to a ProcessDesign in the role process.⌋*(RS_MANI_00005, RS_MANI_00061)*

[constr_10004]{DRAFT} **Consistency of DiagnosticServiceGenericMapping for PortPrototype typed by DiagnosticDataIdentifierGenericInterface** ⌈If a DiagnosticWriteDataByIdentifier and a DiagnosticReadDataByIdentifier exist that refer to the same DiagnosticDataIdentifier and no DiagnosticServiceDataIdentifierPortMapping is used, then two DiagnosticServiceGenericMappings shall exist refer to the same PortPrototype typed by DiagnosticDataIdentifierGenericInterface.⌋*()*

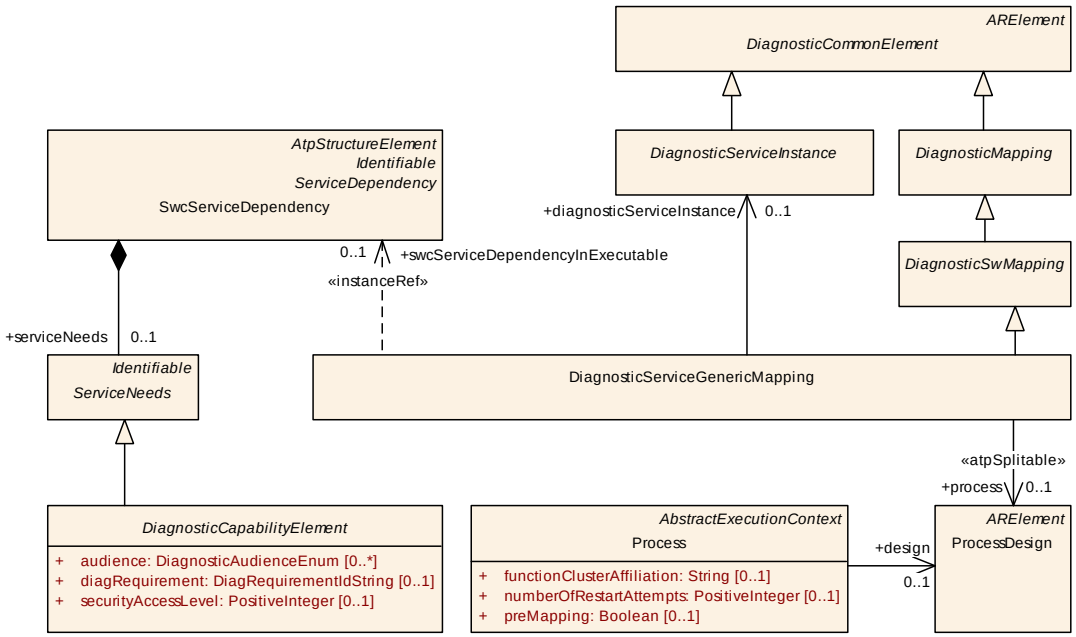


Figure 4.13: Modeling of the diagnostic generic mapping

[constr_1765]{DRAFT} **Diagnostic Services eligible for DiagnosticServiceGenericMapping** ⌈The allowed combinations of DiagnosticPortInterface and ServiceNeeds that are eligible for the utilization of DiagnosticServiceGenericMapping are listed in Table 4.21.⌋*()*

DiagnosticPortInterface	ServiceNeeds
DiagnosticRoutineInterface	DiagnosticRoutineNeeds
DiagnosticGenericUdsInterface	DiagnosticGenericUdsNeeds
DiagnosticRoutineGenericInterface	DiagnosticRoutineNeeds
DiagnosticDataIdentifierGenericInterface	DiagnosticValueNeeds
DiagnosticUploadInterface	DiagnosticUploadDownloadNeeds
DiagnosticDownloadInterface	DiagnosticUploadDownloadNeeds
DiagnosticEcuResetInterface	DiagnosticControlNeeds

Table 4.21: Relation between PortInterface and ServiceNeeds for the DiagnosticServiceGenericMapping

Class	**DiagnosticServiceGenericMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	This meta-class represents the ability to implement a generic generic mapping for select diagnostics services on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticServiceMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnostic ServiceInstance	DiagnosticService Instance	0..1	ref	Reference to the ServiceInstance mapped to a SWC service port. **Tags:**atp.Status=draft
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
swcService DependencyIn Executable	SwcService Dependency	0..1	iref	This aggregation allows for the usage of the Diagnostic ServiceGenericMapping on the AUTOSAR adaptive platform. **Tags:**atp.Status=draft **InstanceRef implemented by:**SwcServiceDependency InExecutableInstanceRef

Table 4.22: DiagnosticServiceGenericMapping

4.1.12 Diagnostic Data Mapping

Disclaimer: The DiagnosticServiceDataMapping is currently not supported as input for the configuration of AUTOSAR Adaptive Diagnostic Management.

[TPS_MANI_01037]{DRAFT} **Diagnostic data mapping on the *AUTOSAR adaptive platform*** ⌈The diagnostic data mapping on the *AUTOSAR adaptive platform* is created by means of meta-class DiagnosticServiceDataMapping that maps a DiagnosticDataElement to a DataPrototype referenced in the role mappedAp-DataElement.⌋*(RS_MANI_00005)*

[TPS_MANI_01060]{DRAFT} **Use cases for the application of `DiagnosticServiceDataMapping`** ⌈`DiagnosticServiceDataMapping` shall only be used where access to data is free of side effects. This is the case for the notifier events of `field`s and, at least with respect to the value, `event`s.⌋*(RS_MANI_00005)*

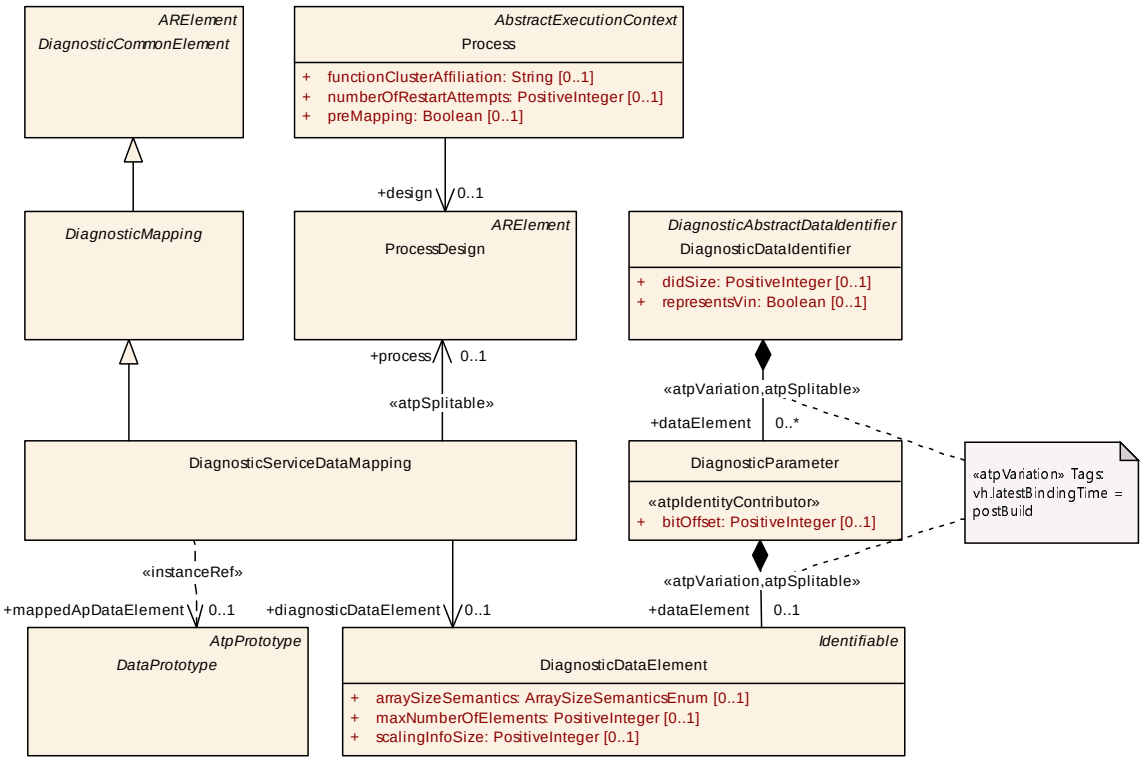


Figure 4.14: Modeling of the diagnostic data mapping

Please note that the `DiagnosticServiceDataMapping` can be applied on models on the *AUTOSAR adaptive platform* because the mapping target is a `DataPrototype` that is aggregated by a `ServiceInterface` in the context of a `PortPrototype`.

In other words, the `DiagnosticServiceDataMapping` applies for the mapping to an `event` or `field`, **or even to an element of** an `event` or `field`.

[constr_1496]{DRAFT} **`DiagnosticServiceDataMapping.mappedApDataElement` shall only refer to specific sub-classes of `DataPrototype`** ⌈A `DiagnosticServiceDataMapping.mappedApDataElement` shall only refer to an `event` or a `field` or a `DataPrototype` owned by an `event` or a `field`.⌋*()*

Please note that the existence of [constr_1496] is a direct consequence of the existence of [TPS_MANI_01060].

In particular, [constr_1496] prevents the creation of a `DiagnosticServiceDataMapping` to a `ArgumentDataPrototype`. In the diagnostic context, `ArgumentDataPrototype` are mainly used in the argument list of the sub-functions of diagnostic routines which are rarely free of side effects.

Class	DiagnosticServiceDataMapping			
Package	M2::AUTOSARTemplates::DiagnosticExtract::ServiceMapping			
Note	This represents the ability to define a mapping of a diagnostic service to a software-component. This kind of service mapping is applicable for the usage of SenderReceiverInterfaces or event/notifier semantics in ServiceInterfaces on the adaptive platform. **Tags:**atp.recommendedPackage=DiagnosticServiceMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
diagnosticData Element	DiagnosticDataElement	0..1	ref	This represents the applicable payload that corresponds to the referenced DataPrototype in the role mappedData Element or (in case of a usage on the adaptive platform) mappedApDataElement.
mappedApData Element	DataPrototype	0..1	iref	This represents the dataElement in the application software of an adaptive AUTOSAR application that is accessed for diagnostic purpose. **Tags:**atp.Status=draft **InstanceRef implemented by:**DataPrototypeIn ExecutableInstanceRef
mappedData Element	DataPrototype	0..1	iref	This represents the dataElement in the application software that is accessed for diagnostic purpose. This role is applicable on the classic platform. **InstanceRef implemented by:**DataPrototypeInSystem InstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft

Table 4.23: DiagnosticServiceDataMapping

Class	DiagnosticDataElement			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents the ability to describe a concrete piece of data to be taken into account for diagnostic purposes.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls the meaning of the value of the array size.
maxNumberOf Elements	PositiveInteger	0..1	attr	The existence of this attribute turns the data instance into an array of data. The attribute determines the size of the array in terms of how many elements the array can take.
scalingInfoSize	PositiveInteger	0..1	attr	Size in bytes of scaling information for the DiagnosticData Element if used with DiagnosticReadScalingDataBy Identifier
swDataDef Props	SwDataDefProps	0..1	aggr	This property allows to specify data definition properties in order to support the definition of e.g. computation formulae and data constraints.

Table 4.24: DiagnosticDataElement

4.2 Diagnostic Clear Condition

On the *AUTOSAR adaptive platform*, a new model element similar DiagnosticEnableCondition is introduced: DiagnosticClearCondition.

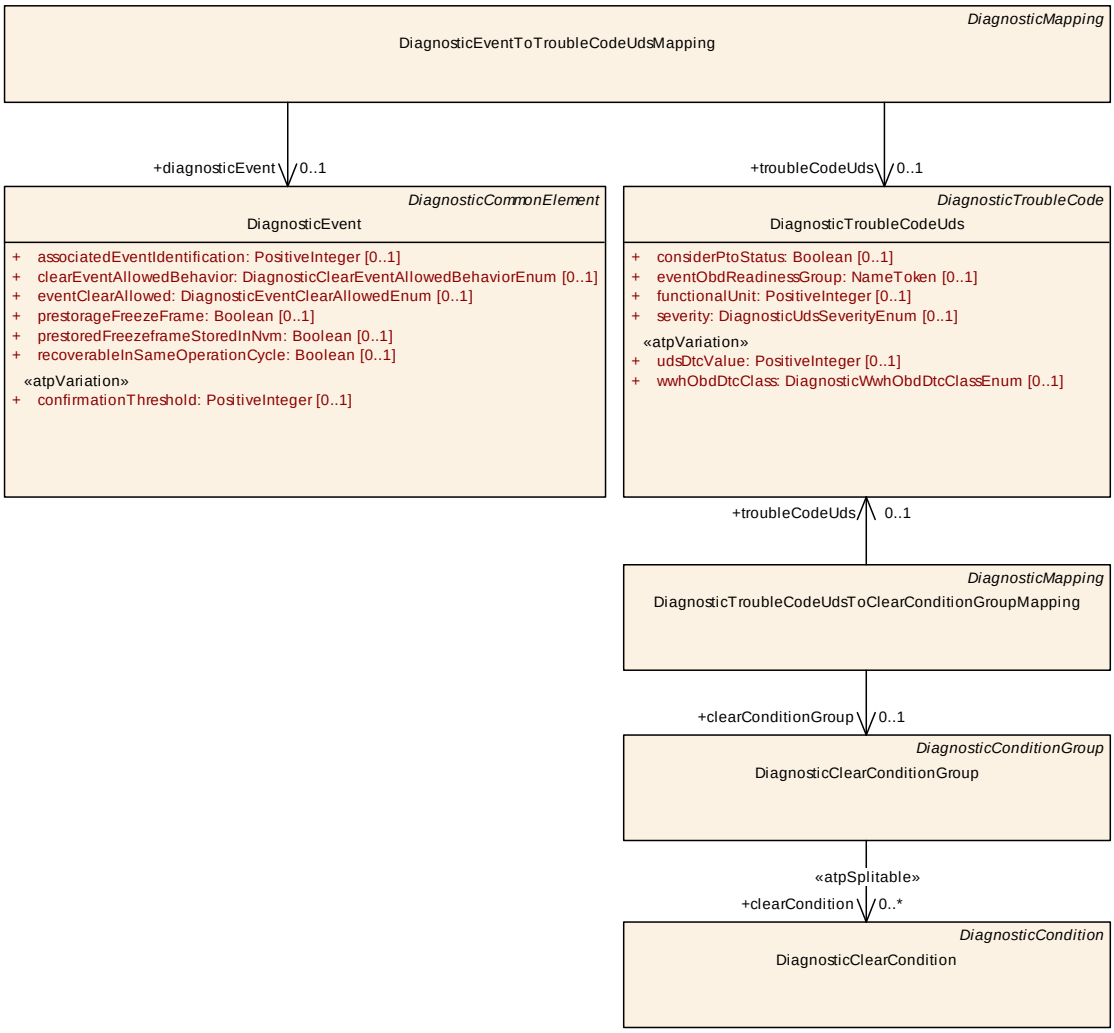


Figure 4.15: Modeling of the diagnostic clear condition

In contrast to DiagnosticEnableCondition, DiagnosticClearCondition is not mapped to a DiagnosticEvent but (via the aggregation by DiagnosticClearConditionGroup) to a DiagnosticTroubleCodeUds.

For this purpose, meta-class DiagnosticTroubleCodeUdsToClearConditionGroupMapping has been defined.

Class	DiagnosticClearCondition			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition			
Note	This meta-class describes a clear condition for diagnostic purposes. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticConditions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticCondition*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.25: DiagnosticClearCondition

Class	DiagnosticClearConditionGroup			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition			
Note	Clear condition group which includes one or several clear conditions. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticClearConditionGroups			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticConditionGroup*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
clearCondition	DiagnosticClear Condition	*	ref	This aggregation represents the collection of Diagnostic ClearConditions that belong to the DiagnosticClear ConditionGroup. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=clearCondition atp.Status=draft

Table 4.26: DiagnosticClearConditionGroup

Class	DiagnosticTroubleCodeUdsToClearConditionGroupMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition			
Note	This meta-class provides the ability to map a DiagnosticClearConditionGroup to a collection of Diagnostic TroubleCodeUds. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
clearCondition Group	DiagnosticClear ConditionGroup	0..1	ref	Thi reference identifies the applicable DiagnosticClear ConditionGroup. **Tags:**atp.Status=draft
troubleCodeUds	DiagnosticTroubleCode Uds	0..1	ref	This reference identifies the DiagnosticTroubleCodeUds that are relevant for the mapping. **Tags:**atp.Status=draft

Table 4.27: DiagnosticTroubleCodeUdsToClearConditionGroupMapping

[constr_1658]{DRAFT} **Number of `DiagnosticTroubleCodeUdsToClearConditionGroupMapping` elements per `DiagnosticTroubleCodeUds`** ⌈The mapping element `DiagnosticTroubleCodeUdsToClearConditionGroupMapping` shall be created no more than once per `DiagnosticTroubleCodeUds`.

If several `DiagnosticTroubleCodeUdsToClearConditionGroupMapping` elements referring to the same `DiagnosticTroubleCodeUds` are defined, then the `Clear Condition Group` mapping shall be regarded as defective.⌋*()*

4.3 Security Access

the implementation of the diagnostics manager on the adaptive platform requires a refined modeling of meta-class `DiagnosticSecurityAccessClass`.

A new attribute named `sharedTimer` is introduced that controls whether a single timer is used for all security access levels or whether the individual levels utilize separate timers respectively.

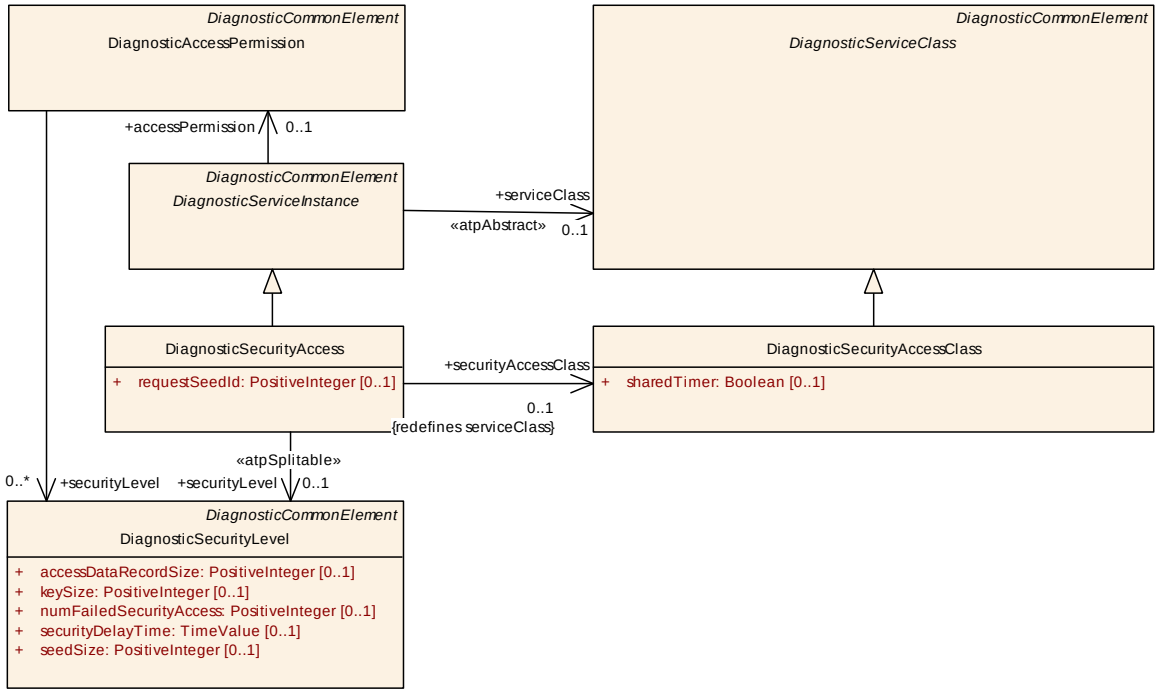


Figure 4.16: Refined modeling of the diagnostic security access

Class	**DiagnosticSecurityAccessClass**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::SecurityAccess
Note	This meta-class contains attributes shared by all instances of the "Security Access" diagnostic service. **Tags:**atp.recommendedPackage=DiagnosticSecurityAccesss

△

Class	**DiagnosticSecurityAccessClass**			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sharedTimer	Boolean	0..1	attr	Switch between separate or single shared timer instance and timer value. • True: use shared timer instance and timer value for all security access levels combined. • False: use separate timer instance and timer values for each security level. **Tags:**atp.Status=draft

Table 4.28: DiagnosticSecurityAccessClass

4.4 DiagnosticProvidedDataMapping

[TPS_MANI_01230]{DRAFT} **Semantics of `DiagnosticProvidedDataMapping`**
⌈The meta-class `DiagnosticProvidedDataMapping` does not seem to fulfill the condition for representing a mapping class because it only has one reference to a `DiagnosticDataElement` in the role `dataElement`.

However, the specific nature of this mapping is that the second element (the `DiagnosticProvidedDataMapping.dataProvider`) that is supposed to take place in the mapping cannot precisely be modeled as a single meta-class.

Therefore, there is no better way than to model the `DiagnosticProvidedDataMapping.dataProvider` by a `NameToken`.⌋*()*

Class	**DiagnosticProvidedDataMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticProvidedDataMapping			
Note	This represents the ability to define the nature of a data access for a DiagnosticDataElement based on a data provider that cannot be modeled explicitly. **Tags:** atp.Status=draft atp.recommendedPackage=DataMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	DiagnosticDataElement	0..1	ref	This represents the DiagnosticDataElement for which the access is further qualified by the DiagnosticProvidedData Mapping.dataProvider. **Tags:**atp.Status=draft
dataProvider	NameToken	1	attr	This represents the ability to further specify the data provider.

Table 4.29: DiagnosticProvidedDataMapping

Please note that the list of standardized values of attribute `DiagnosticProvided-DataMapping.dataProvider` is defined in the SWS Diagnostics [20].

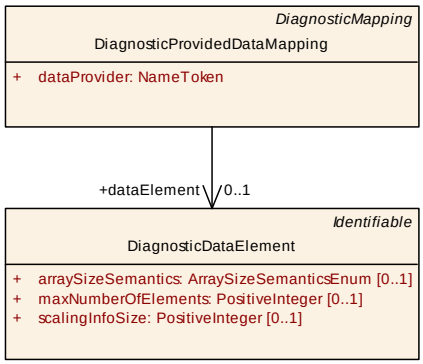


Figure 4.17: Modeling of `DiagnosticProvidedDataMapping`

5 System Design

5.1 Overview

A typical vehicle will most likely be equipped with ECUs developed on the AUTOSAR classic platform and ECUs developed on the AUTOSAR adaptive platform. The system design for the entire vehicle has therefore to cover all these ECUs.

The AUTOSAR model description supports the system design with the possibility to describe Software Components of both AUTOSAR Platforms that will be used in a System and even allows to indicate the service oriented communication between them if possible.

Especially when it come to the description of the communication behavior of AUTOSAR classic and adaptive ECUs in a harmonized way the notion of a System Design becomes a special focus point.

All the system design aspects have in common that they have to cope with both, AUTOSAR classic and adaptive. The basic design aspects of such interdisciplinary systems have to be already available in the AUTOSAR classic modeling approach because otherwise they would not be available to both worlds.

Thus it is straight forward to take the existing meta-class `System` as the starting point for the modeling of such mixed systems.

Class	**System**			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	The top level element of the System Description. **Tags:**atp.recommendedPackage=Systems			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
fibexElement	FibexElement	*	ref	Reference to ASAM FIBEX elements specifying Communication and Topology. All Fibex Elements used within a System Description shall be referenced from the System Element. atpVariation: In order to describe a product-line, all Fibex Elements can be optional. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
interpolation Routine MappingSet	InterpolationRoutine MappingSet	*	ref	This reference identifies the InterpolationRoutineMapping Sets that are relevant in the context of the enclosing System.

▽

△

Class	**System**			
mapping	SystemMapping	*	aggr	Aggregation of all mapping aspects relevant in the System Description. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=mapping.shortName, mapping.variation Point.shortLabel vh.latestBindingTime=postBuild
pncVector Length	PositiveInteger	0..1	attr	Length of the partial networking request release information vector (in bytes).
pncVectorOffset	PositiveInteger	0..1	attr	Absolute offset (with respect to the NM-PDU) of the partial networking request release information vector that is defined in bytes as an index starting with 0.
rootSoftware Composition	RootSwComposition Prototype	0..1	aggr	Aggregation of the root software composition, containing all software components in the System in a hierarchical structure. This element is not required when the System description is used for a network-only use-case. atpVariation: The RootSwCompositionPrototype can vary. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=rootSoftwareComposition.shortName, root SoftwareComposition.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime
systemVersion	RevisionLabelString	1	attr	Version number of the System Description.

Table 5.1: System

[constr_3366]{DRAFT} **System `category` for a system design description with Adaptive Platform and Classic Platform content** ⌈The `System` element that contains design artifacts that are relevant for the Adaptive Platform and Classic Platform shall have the `category` `SYSTEM_DESIGN_DESCRIPTION`.⌋*()*

There are use cases to exchange parts of such a `SYSTEM_DESIGN_DESCRIPTION` between different developer parties and therefore further system categories are supported by AUTOSAR.

A common approach is for example that the OEM provides a basis for designing an ECU, which is later advanced by the supplier. Therefore Classic AUTOSAR supports `System` categories like `ECU_EXTRACT` or `ECU_SYSTEM_DESCRIPTION` that have only a single ECU in scope.

Adaptive AUTOSAR is using the same approach. If an OEM wants to provide design artifacts that are relevant for the configuration of a single `Machine` all unnecessary information is stripped from the `System` with `category` `SYSTEM_DESIGN_DESCRIPTION` and a definition of the subsystem is provided.

[TPS_MANI_01274]{DRAFT} **System `category` for a design description that has one single Adaptive `Machine` in scope** ⌈The `System` element that contains design artifacts that are relevant for a single Adaptive `Machine` shall have the `category` `MACHINE_DESIGN_EXTRACT`.⌋*()*

[constr_3421]{DRAFT} **Fibex elements applicable for a `System` of `category` `MACHINE_DESIGN_EXTRACT`** ⌈A `System` with the `category` `MACHINE_DESIGN_EXTRACT` is allowed to reference the following `fibexElement`s:

- `CommunicationCluster`
- `MachineDesign`
- `GlobalTimeDomain`
- `NmConfig`
- `DltMessageCollectionSet`
- `SystemMapping` that is allowed to contain only a `PncMapping`

⌋()

5.2 Specification of Communication System Structure

When the communication interaction is designed for a vehicle system the focus is put on the network and the connected ECUs. Whether a specific ECU connected to the network is implemented using AUTOSAR classic or AUTOSAR adaptive does not influence the major communication design.

But of course, it is essential from a car manufacturer point of view whether a specific ECU will be implemented using AUTOSAR classic or adaptive. Thus, already on system design level there is a need to specify the AUTOSAR Platform kind which shall be used to implement an ECU.

In AUTOSAR classic the element `EcuInstance` is used to define one ECU in the system design.

In AUTOSAR adaptive the element `Machine` is an entity which already represents a specific ECU Implementation with dedicated configurations for e.g. `Processor`s, `ProcessorCore`s.

The `Machine` is a model entity which is not in the focus of communication designers and should not be used during system design.

Therefore, the `MachineDesign` has been introduced in order to allow the communication system designer to define a placeholder for an adaptive ECU in the scope of the `System` (the `MachineDesign` corresponds to the `EcuInstance` of AUTOSAR classic).

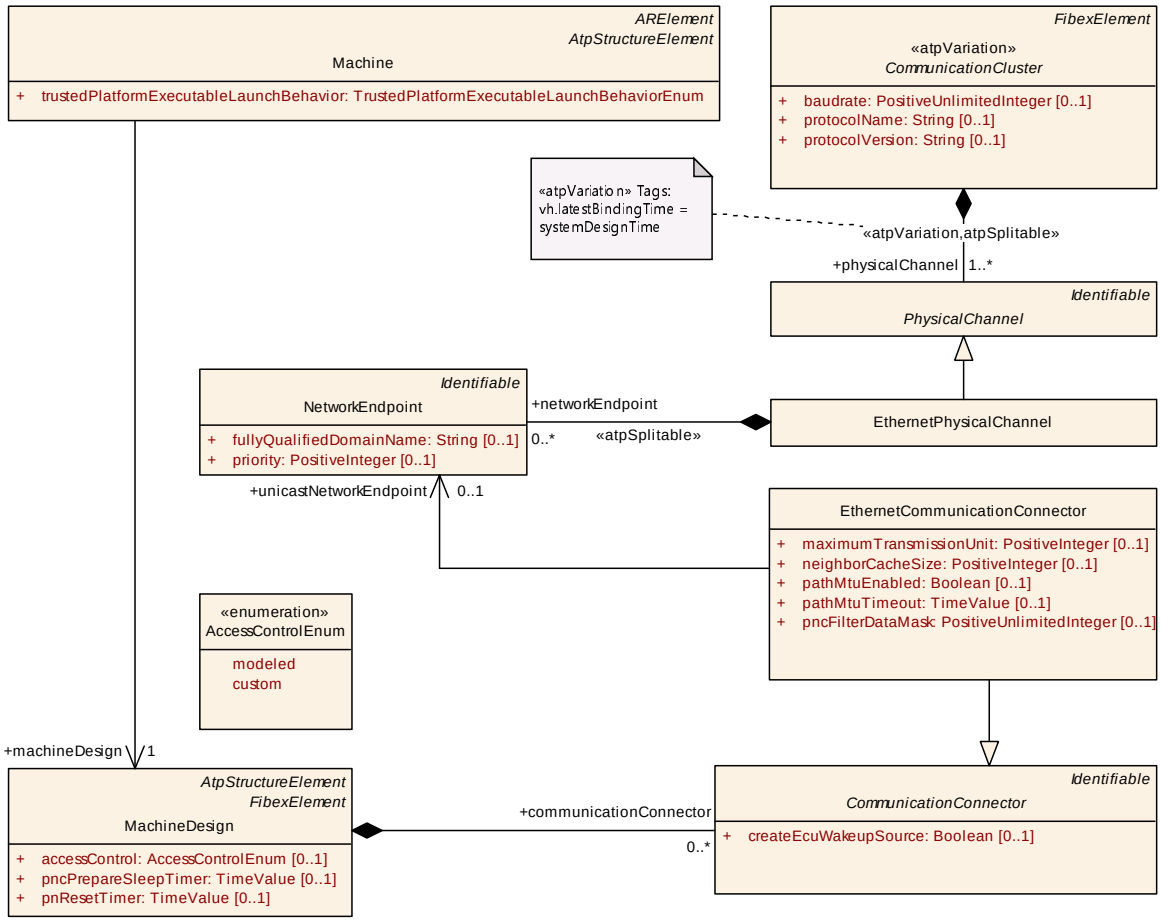


Figure 5.1: MachineDesign

Class	**MachineDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This meta-class represents the ability to define requirements on a Machine in the context of designing a system. **Tags:** atp.Status=draft atp.recommendedPackage=MachineDesigns			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
accessControl	AccessControlEnum	0..1	attr	This attribute defines how the access restriction to the Service Instance is defined.
communication Connector	Communication Connector	*	aggr	This aggregation defines the network connection of the machine. **Tags:**atp.Status=draft
ethIpProps	EthIpProps	0..1	ref	Maschine specific IP attributes. **Tags:**atp.Status=draft
pncPrepare SleepTimer	TimeValue	0..1	attr	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.

△

Class	**MachineDesign**			
pnResetTimer	TimeValue	0..1	attr	Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests.
serviceDiscoverConfig	ServiceDiscoveryConfiguration	*	aggr	Set of service discovery configuration settings that are defined on the machine for individual Communication Connectors. **Tags:**atp.Status=draft
tcpIpIcmpProps	EthTcpIpIcmpProps	0..1	ref	Machine specific ICMP (Internet Control Message Protocol) attributes **Tags:**atp.Status=draft
tcpIpProps	EthTcpIpProps	0..1	ref	Machine specific TcpIp Stack attributes. **Tags:**atp.Status=draft

Table 5.2: MachineDesign

[TPS_MANI_03209]{DRAFT} **The meaning of `MachineDesign.accessControl`**
⌈The `MachineDesign.accessControl` defines whether the access control is defined by AUTOSAR means in the Application Design with `receiverIntent` (see [TPS_MANI_01106]) and `senderIntent` (see [TPS_MANI_01107]) or by a custom lists that are created by a non-AUTOSAR process.⌋*(RS_MANI_00034)*

Enumeration	**AccessControlEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment
Note	This enumeration describes the options for the definition of access restriction to resources. **Tags:**atp.Status=draft
Literal	*Description*
custom	The access restriction to the resource is defined by a non-AUTOSAR process. **Tags:**atp.EnumerationLiteralIndex=1
modeled	The access restriction to the resource is modeled in the AUTOSAR Application Design model or the AUTOSAR Deployment model. **Tags:**atp.EnumerationLiteralIndex=0

Table 5.3: AccessControlEnum

5.2.1 Network connection

One of the most prominent information defined in the context of the `MachineDesign` is the network connectivity. Since the *AUTOSAR adaptive platform* focuses on the usage of Ethernet for communication, this boils down to the specification of IP addresses.

Specifically, the basic definition of the connectivity of a `MachineDesign` is created by aggregating the abstract base-class `CommunicationConnector` in the role `communicationConnector`. The specific subclass of `CommunicationConnector` that is used in this context is the `EthernetCommunicationConnector`.

The EthernetCommunicationConnector is used to connect the MachineDesign with a VLAN that is represented in AUTOSAR by a EthernetPhysicalChannel that is part of an EthernetCluster.

Class	***PhysicalChannel*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	This element represents a physical connection (in case of CAN, FlexRay, LIN) or a logical connection (VLAN in case of Ethernet) between communicating devices.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractCanPhysicalChannel*, EthernetPhysicalChannel, FlexrayPhysicalChannel, LinPhysicalChannel, UserDefinedPhysicalChannel			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 5.4: PhysicalChannel

Class	<<atpVariation>> **EthernetCluster**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Ethernet-specific cluster attributes. **Tags:**atp.recommendedPackage=CommunicationClusters			
Base	*ARObject*, *CollectableElement*, *CommunicationCluster*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
couplingPortConnection	CouplingPortConnection	*	aggr	Specification of connections between CouplingElements and EcuInstances. Note: This atpSplitable property has no atp.Splitkey due to atpVariation (PropertySetPattern). **Stereotypes:** atpSplitable; atpVariation **Tags:**vh.latestBindingTime=postBuild
couplingPortStartupActiveTime	TimeValue	0..1	attr	The attribute specifies the time in second a coupling port is switched on to enable the host ECU (ECU that maintains an Ethernet switch) to listen to the network for potential network management requests.
couplingPortSwitchoffDelay	TimeValue	0..1	attr	Switch off delay for CouplingPorts in seconds. It denotes the delay of switching off couplingPorts after the request to switch off a couplingPort was issued. (e.g. switch off of Ethernet switch ports).
macMulticastGroup	MacMulticastGroup	*	aggr	MacMulticastGroup that is defined for the Subnet (EthernetCluster).

Table 5.5: EthernetCluster

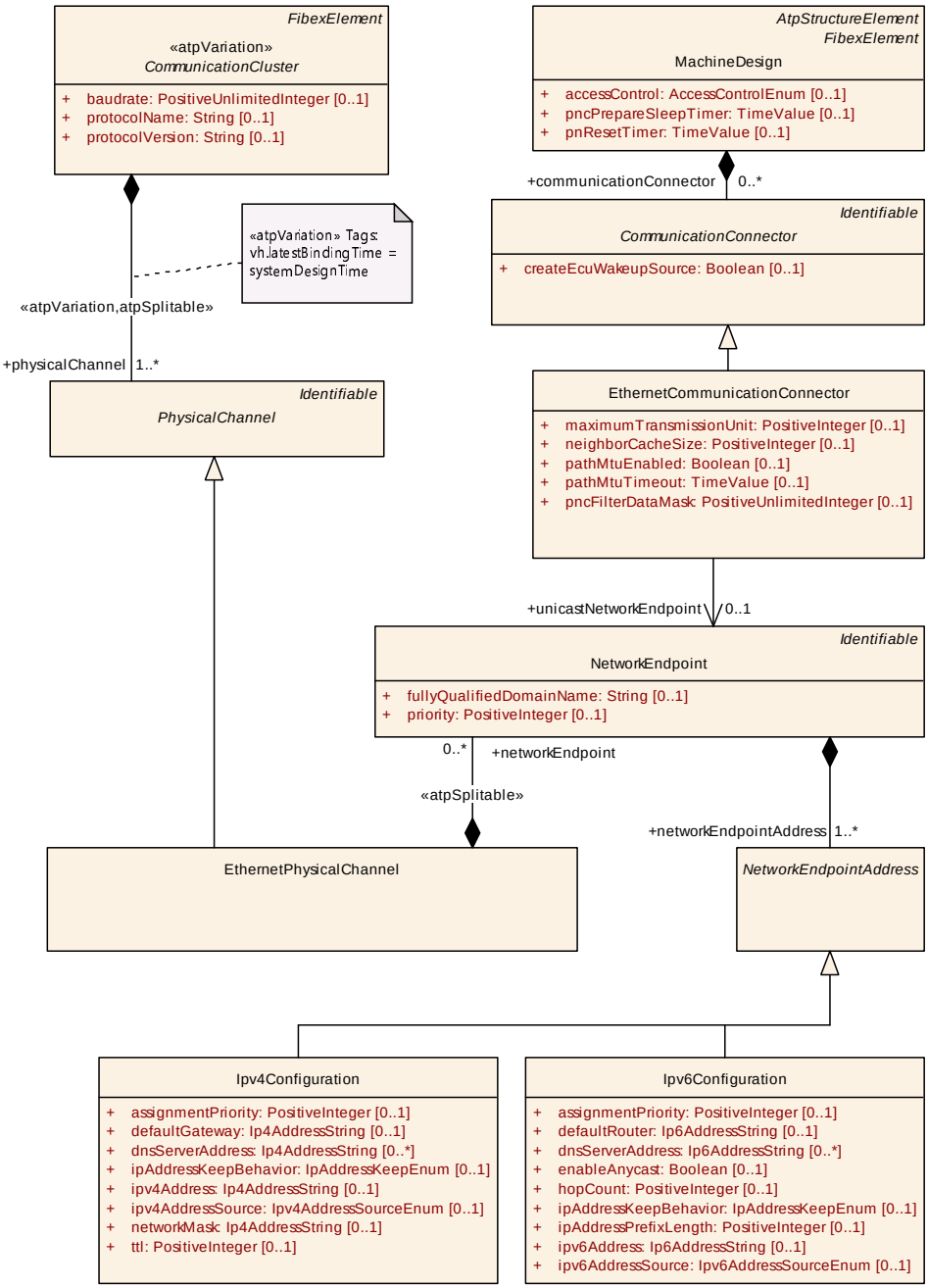


Figure 5.2: Network connection of a MachineDesign

[constr_3320]{DRAFT} **Aggregation of `CommunicationConnector` by `MachineDesign`** ⌈Meta-Class `MachineDesign` shall only aggregate `EthernetCommunicationConnector`s in the role `communicationConnector`. No other subclass of `CommunicationConnector` shall appear in this aggregation.⌋*()*

The canonical way to specify an IP address is the modeling of a `NetworkEndpoint`, referenced from an `EthernetCommunicationConnector` that is aggregated by `MachineDesign` in the role `communicationConnector`.

In addition to the IP address, the `NetworkEndpoint` may have a *Fully Qualified Domain Name* and a priority.

Class	**NetworkEndpoint**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	The network endpoint defines the network addressing (e.g. IP-Address or MAC multicast address).			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
fullyQualified DomainName	String	0..1	attr	Defines the fully qualified domain name (FQDN) e.g. some.example.host.
ipSecConfig	IPSecConfig	0..1	aggr	Optional IPSec configuration that provides security services for IP packets.
network Endpoint Address	NetworkEndpoint Address	1..*	aggr	Definition of a Network Address. **Tags:**xml.name Plural=NETWORK-ENDPOINT-ADDRESSES
priority	PositiveInteger	0..1	attr	Defines the frame priority where values from 0 (best effort) to 7 (highest) are allowed.

Table 5.6: NetworkEndpoint

More precisely, the particular IP address is configured by means of the aggregation of `Ipv4Configuration` or `Ipv6Configuration` in the role `networkEndpointAddress`.

The `NetworkEndpoint` is aggregated by the `EthernetPhysicalChannel` that in turn is aggregated by the `EthernetCluster`.

[TPS_MANI_03052]{DRAFT} **Static IPv4 configuration** ⌈If the value of attribute `ipv4AddressSource` of meta-class `Ipv4Configuration` is set to `Ipv4AddressSourceEnum.fixed` then the `ipv4Address` defines the static IPv4 Address.⌋*(RS_MANI_00018)*

Class	**Ipv4Configuration**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Internet Protocol version 4 (IPv4) configuration.			
Base	*ARObject*, *NetworkEndpointAddress*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
assignment Priority	PositiveInteger	0..1	attr	Priority of assignment (1 is highest). If a new address from an assignment method with a higher priority is available, it overwrites the IP address previously assigned by an assignment method with a lower priority.
defaultGateway	Ip4AddressString	0..1	attr	IP address of the default gateway.
dnsServer Address	Ip4AddressString	*	attr	IP addresses of preconfigured DNS servers. **Tags:**xml.namePlural=DNS-SERVER-ADDRESSES
ipAddressKeep Behavior	IpAddressKeepEnum	0..1	attr	Defines the lifetime of a dynamically fetched IP address.
ipv4Address	Ip4AddressString	0..1	attr	IPv4 Address. Notation: 255.255.255.255. The IP Address shall be declared in case the ipv4AddressSource is FIXED and thus no auto-configuration mechanism is used.
ipv4Address Source	Ipv4AddressSource Enum	0..1	attr	Defines how the node obtains its IP address.

▽

△

Class	**Ipv4Configuration**			
networkMask	Ip4AddressString	0..1	attr	Network mask. Notation 255.255.255.255
ttl	PositiveInteger	0..1	attr	Lifespan of data (0..255). The purpose of the TimeToLive field is to avoid a situation in which an undeliverable datagram keeps circulating on a system.

Table 5.7: Ipv4Configuration

Enumeration	**Ipv4AddressSourceEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology
Note	Defines how the node obtains its IPv4-Address.
Literal	*Description*
autoIp	AutoIP is used to dynamically assign IP addresses at device startup. **Tags:**atp.EnumerationLiteralIndex=0
autoIp_doip	Linklocal IPv4 Address Assignment using DoIP Parameters **Tags:**atp.EnumerationLiteralIndex=2
dhcpv4	DHCP is a service for the automatic IP configuration of a client. **Tags:**atp.EnumerationLiteralIndex=3
fixed	The IP Address shall be declared manually. **Tags:**atp.EnumerationLiteralIndex=4

Table 5.8: Ipv4AddressSourceEnum

[TPS_MANI_03053]{DRAFT} **Static IPv6 configuration** ⌈If the value of attribute `ipv6AddressSource` of meta-class `Ipv6Configuration` is set to `Ipv6AddressSourceEnum.fixed` then the `ipv6Address` defines the static IPv6 Address.⌋*(RS_MANI_00018)*

Class	**Ipv6Configuration**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Internet Protocol version 6 (IPv6) configuration.			
Base	*ARObject*, *NetworkEndpointAddress*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
assignment Priority	PositiveInteger	0..1	attr	Priority of assignment (1 is highest). If a new address from an assignment method with a higher priority is available, it overwrites the IP address previously assigned by an assignment method with a lower priority.
defaultRouter	Ip6AddressString	0..1	attr	IP address of the default router.
dnsServer Address	Ip6AddressString	*	attr	IP addresses of pre configured DNS servers. **Tags:**xml.namePlural=DNS-SERVER-ADDRESSES
enableAnycast	Boolean	0..1	attr	This attribute is used to enable anycast addressing (i.e. to one of multiple receivers).
hopCount	PositiveInteger	0..1	attr	The distance between two hosts. The hop count n means that n gateways separate the source host from the destination host (Range 0..255)
ipAddressKeep Behavior	IpAddressKeepEnum	0..1	attr	Defines the lifetime of a dynamically fetched IP address.

▽

△

Class	**Ipv6Configuration**			
ipAddressPrefix Length	PositiveInteger	0..1	attr	IPv6 prefix length defines the part of the IPv6 address that is the network prefix.
ipv6Address	Ip6AddressString	0..1	attr	IPv6 Address. Notation: FFFF:...:FFFF. The IP Address shall be declared in case the ipv6AddressSource is FIXED and thus no auto-configuration mechanism is used.
ipv6Address Source	Ipv6AddressSource Enum	0..1	attr	Defines how the node obtains its IP address.

Table 5.9: Ipv6Configuration

Enumeration	**Ipv6AddressSourceEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology
Note	Defines how the node obtains its IPv6-Address.
Literal	***Description***
dhcpv6	DHCP is a service for the automatic IP configuration of a client. **Tags:**atp.EnumerationLiteralIndex=0
fixed	The IP Address shall be declared manually. **Tags:**atp.EnumerationLiteralIndex=1
linkLocal	LinkLocal is intended only for communications within the segment of a local network (a link) or a point-to-point connection that a host is connected to. **Tags:**atp.EnumerationLiteralIndex=2
linkLocal_doip	Linklocal IPv6 Address Assignment using DoIP Parameters **Tags:**atp.EnumerationLiteralIndex=3
router Advertisement	IPv6 Stateless Autoconfiguration. **Tags:**atp.EnumerationLiteralIndex=4

Table 5.10: Ipv6AddressSourceEnum

Enumeration	**IpAddressKeepEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology
Note	Defines the behavior after a dynamic IP address has been assigned.
Literal	***Description***
forget	After a dynamic IP address has been assigned just use it for this session. **Tags:**atp.EnumerationLiteralIndex=0
storePersistently	After a dynamic IP address has been assigned store the address persistently. **Tags:**atp.EnumerationLiteralIndex=1

Table 5.11: IpAddressKeepEnum

Please note that there is also the possibility to describe a `MacMulticastConfiguration` as `NetworkEndpointAddress` in addition to `Ipv4Configuration` and `Ipv6Configuration` in a system topology description. This may be useful for description of endpoints that are not based on IP, e.g. for streaming protocols like AVB (Audio Video Bridging). But please note that there is no foundation software or ara::com support for such `MacMulticastConfiguration` `NetworkEndpoint`s in Adaptive Autosar. For SOME/IP communication such `NetworkEndpoint`s are excluded by [constr_3288].

5.2.1.1 Support of 10BASE-T1S Network Topologies

Please note that 10BASE-T1S network topology description is supported in the System Design with a `CouplingPortConnection` that points with the `nodePort` reference to `CouplingPort`s that represent the 10Base-T1S PHYs connected to the network.

More details about the modeling of 10BASE-T1S networks can be found in the System Template [17]. Since the same modeling approach is used in the Classic Platform and the Adaptive Platform the detailed description of the meta-model available in the System Template is not repeated in this specification.

5.2.2 TcpIp stack configuration properties

The `MachineDesign` references the following elements and allows to set Machine specific TcpIp stack configuration options in the System Design:

- `EthIpProps` - used to configure IPv4 and IPv6
- `EthTcpIpProps` - used to configure TCP and UDP
- `EthTcpIpIcmpProps` - used to configure ICMP

Please note that the System Template [17] defines constraints for the usage of `EthIpProps`, `EthTcpIpProps`, `EthTcpIpIcmpProps`. These constraints are also valid if the `MachineDesign` references these elements.

5.2.2.1 IP configuration properties

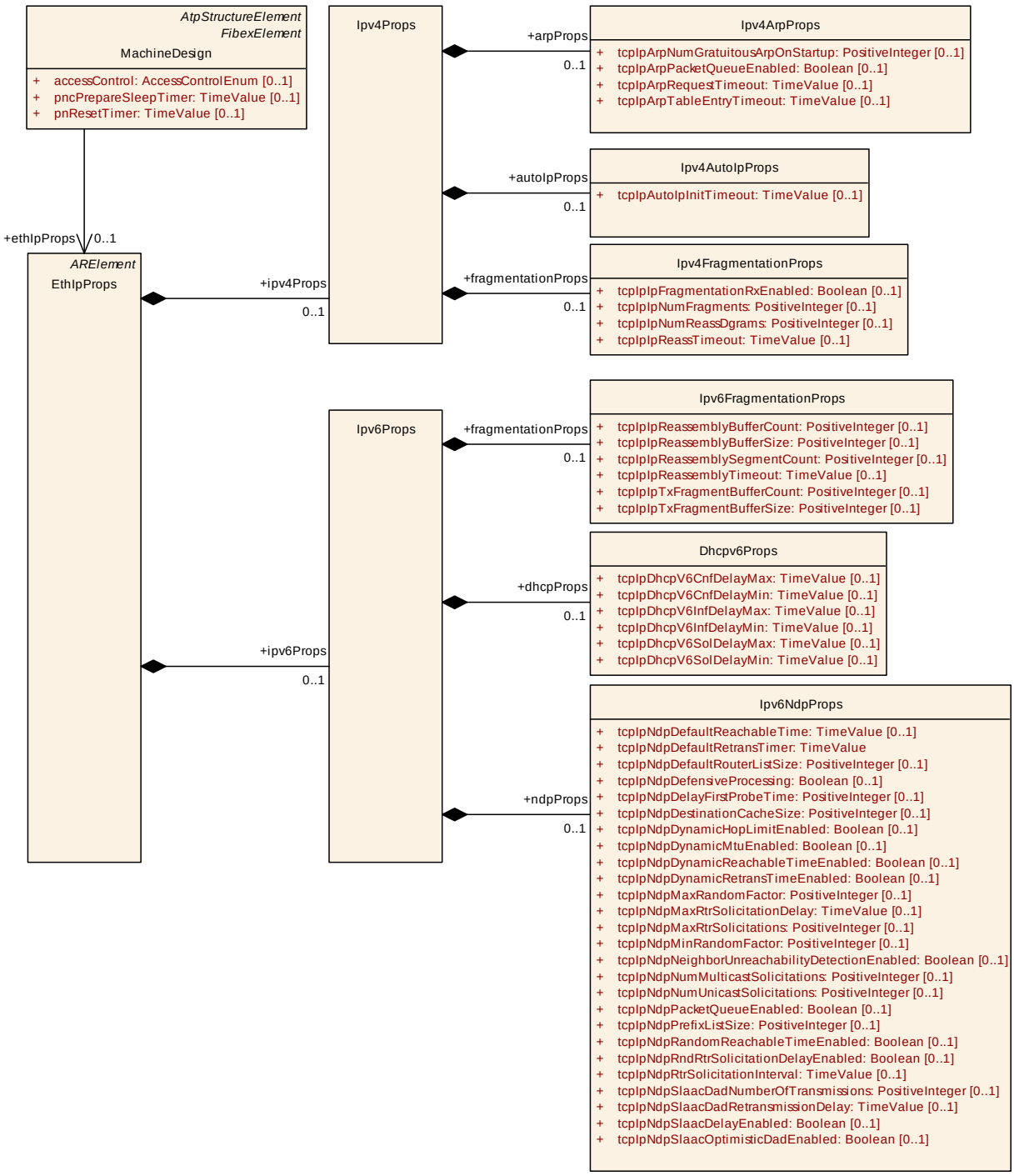


Figure 5.3: Machine specific IP configuration options

Class	EthIpProps			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class is used to configure the Machine specific IP attributes. **Tags:**atp.recommendedPackage=EthIpProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ipv4Props	Ipv4Props	0..1	aggr	Configuration options for IPv4.
ipv6Props	Ipv6Props	0..1	aggr	Configuration options for IPv6.

Table 5.12: EthIpProps

Class	Ipv4Props			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class specifies the configuration options for IPv4.			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
arpProps	Ipv4ArpProps	0..1	aggr	Configuration properties for the ARP (Address Resolution Protocol).
autoIpProps	Ipv4AutoIpProps	0..1	aggr	Configuration options for Auto-IP (automatic private IP addressing).
fragmentation Props	Ipv4Fragmentation Props	0..1	aggr	Configuration options for IPv4 packet fragmentation/ reassembly.

Table 5.13: Ipv4Props

Class	Ipv4ArpProps			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Specifies the configuration options for the ARP (Address Resolution Protocol).			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
tcpIpArpNum GratuitousArp OnStartup	PositiveInteger	0..1	attr	This attribute specifies the number of gratuitous ARP replies which shall be sent on assignment of a new IP address.
tcpIpArpPacket QueueEnabled	Boolean	0..1	attr	This attribute enables (TRUE) or disables (FALSE) support of the ARP Packet Queue according to IETF RFC 1122, section 2.3.2.2.
tcpIpArp Request Timeout	TimeValue	0..1	attr	This attribute specifies a timeout in seconds for the validity of ARP requests. After the transmission of an ARP request the TcpIp shall skip the transmission of any further ARP requests to the same destination within a duration of tcpIpArpRequestTimeout seconds. (IETF RFC 1122, section 2.3.2.1).
tcpIpArpTable EntryTimeout	TimeValue	0..1	attr	This attribute specifies the timeout in seconds after which an unused ARP entry is removed.

Table 5.14: Ipv4ArpProps

Class	**Ipv4AutoIpProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Specifies the configuration options for Auto-IP (automatic private IP addressing).			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpAutoIpInit Timeout	TimeValue	0..1	attr	This attribute specifies the time in seconds Auto-IP waits at startup, before beginning with ARP probing. This delay is used to give DHCP time to acquire a lease in case a DHCP server is present.

Table 5.15: Ipv4AutoIpProps

Class	**Ipv4FragmentationProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Specifies the configuration options for IPv4 packet fragmentation/reassembly.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpIp Fragmentation RxEnabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support for reassembling of incoming datagrams that are fragmented according to IETF RFC 815 (IP Datagram Reassembly Algorithms).
tcpIpIpNum Fragments	PositiveInteger	0..1	attr	Specifies the maximum number of IP fragments per datagram.
tcpIpIpNum ReassDgrams	PositiveInteger	0..1	attr	Specifies the maximum number of fragmented IP datagrams that can be reassembled in parallel.
tcpIpIpReass Timeout	TimeValue	0..1	attr	Specifies the timeout in [s] after which an incomplete datagram gets discarded.

Table 5.16: Ipv4FragmentationProps

Class	**Ipv6Props**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class specifies the configuration options for IPv6.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dhcpProps	Dhcpv6Props	0..1	aggr	Configuration properties for DHCPv6.
fragmentation Props	Ipv6Fragmentation Props	0..1	aggr	Configuration properties for IPv6 packet fragmentation/ reassembly.
ndpProps	Ipv6NdpProps	0..1	aggr	Configuration properties for the Neighbor Discovery Protocol for IPv6.

Table 5.17: Ipv6Props

Class	**Ipv6FragmentationProps**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology
Note	This meta-class specifies the configuration options for IPv6 packet fragmentation/reassembly.
Base	*ARObject*

▽

△

Class	**Ipv6FragmentationProps**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpIp Reassembly BufferCount	PositiveInteger	0..1	attr	Number of buffers that can be used for fragment reassembly. In case of a reassembly error or if not all fragments are received in time this buffer will be blocked until the specified "Fragment Reassembly Timeout" has been exceeded. A value of 0 disables fragment reassembly.
tcpIpIp Reassembly BufferSize	PositiveInteger	0..1	attr	Size of each fragment tx buffer in bytes.
tcpIpIp Reassembly SegmentCount	PositiveInteger	0..1	attr	Specifies the maximum number of consecutive data segments that can be managed in each reassembly buffer. If all fragments are received in order, only one segment will be needed. To deal with fragments received out of order this value should be configured bigger than 1.
tcpIpIp Reassembly Timeout	TimeValue	0..1	attr	Specifies the timeout in seconds after which an incomplete datagram gets discarded.
tcpIpIpTx FragmentBuffer Count	PositiveInteger	0..1	attr	These buffers will be used if the IpV6 receives packets from the upper layer that do not fit into the MTU and thus must be fragmented. A value of 0 disables tx fragmentation.
tcpIpIpTx FragmentBuffer Size	PositiveInteger	0..1	attr	Size of each fragment tx buffer in bytes.

Table 5.18: Ipv6FragmentationProps

Class	**Dhcpv6Props**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class specifies the configuration options for DHCPv6.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpDhcp V6CnfDelayMax	TimeValue	0..1	attr	Maximum delay in seconds before sending the first Confirm message. If this value is bigger than the previous minimum delay value a random delay will be chosen from the interval.
tcpIpDhcp V6CnfDelayMin	TimeValue	0..1	attr	Minimum delay in seconds before the first Confirm message will be sent.
tcpIpDhcpV6Inf DelayMax	TimeValue	0..1	attr	Maximum delay in seconds before sending the first Information Request message. If this value is bigger than the previous minimum delay value a random delay will be chosen from the interval.
tcpIpDhcpV6Inf DelayMin	TimeValue	0..1	attr	Minimum delay (s) before the first Information Request message will be sent.
tcpIpDhcpV6Sol DelayMax	TimeValue	0..1	attr	Maximum delay in seconds before sending the first Solicit message. If this value is bigger than the previous minimum delay value a random delay will be chosen from the interval.
tcpIpDhcpV6Sol DelayMin	TimeValue	0..1	attr	Minimum delay (s) before the first Solicit message will be sent.

Table 5.19: Dhcpv6Props

Class	**Ipv6NdpProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class specifies the configuration options for the Neighbor Discovery Protocol for IPv6.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpNdpDefault ReachableTime	TimeValue	0..1	attr	Configuration of the ReachableTime (s) specified in [RFC4861 6.3.2. Host Variables].
tcpIpNdpDefault RetransTimer	TimeValue	1	attr	Configures the default value (s) for the RetransTimer variable specified in [RFC4861 6.3.2. Host Variables].
tcpIpNdpDefault RouterListSize	PositiveInteger	0..1	attr	Maximum number of default router entries.
tcpIpNdp Defensive Processing	Boolean	0..1	attr	If enabled the NDP shall only process Neighbor Advertisements which are received in reaction to a previously transmitted Neighbor Solicitation as well as skipping updates to the Neighbor Cache based on received Neighbor Solicitations. If disabled all Neighbor Advertisements and Solicitations shall be processed as specified in RFC4861.
tcpIpNdpDelay FirstProbeTime	PositiveInteger	0..1	attr	Delay before sending the first NUD probe in (s).
tcpIpNdp Destination CacheSize	PositiveInteger	0..1	attr	Maximum number of entries in the destination cache.
tcpIpNdp DynamicHop LimitEnabled	Boolean	0..1	attr	If enabled the default hop limit may be reconfigured based on received Router Advertisements.
tcpIpNdp DynamicMtu Enabled	Boolean	0..1	attr	Allow dynamic reconfiguration of link MTU via Router Advertisements.
tcpIpNdp Dynamic ReachableTime Enabled	Boolean	0..1	attr	If enabled the default Reachable Time value may be reconfigured based on received Router Advertisements.
tcpIpNdp Dynamic RetransTime Enabled	Boolean	0..1	attr	If enabled the default Retransmit Timer value may be reconfigured based on received Router Advertisements.
tcpIpNdpMax RandomFactor	PositiveInteger	0..1	attr	Maximum random factor used for randomization
tcpIpNdpMaxRtr Solicitation Delay	TimeValue	0..1	attr	Maximum delay before the first Router Solicitation will be sent after interface initialization in (s).
tcpIpNdpMaxRtr Solicitations	PositiveInteger	0..1	attr	Maximum number of Router Solicitations that will be sent before the first Router Advertisement has been received.
tcpIpNdpMin RandomFactor	PositiveInteger	0..1	attr	Minimum random factor used for randomization
tcpIpNdp Neighbor Unreachability Detection Enabled	Boolean	0..1	attr	Neighbor Unreachability Detection is used to remove unused entries from the neighbor cache. This feature is a basic feature of NDP and should be turned on.
tcpIpNdpNum Multicast Solicitations	PositiveInteger	0..1	attr	Maximum number of multicast solicitations that will be sent when performing address resolution.
tcpIpNdpNum Unicast Solicitations	PositiveInteger	0..1	attr	Maximum number of unicast solicitations that will be sent when performig Neighbor Unreachability Detection.

△

Class	**Ipv6NdpProps**			
tcpIpNdpPacket QueueEnabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of a NDP Packet Queue according to IETF RFC 4861, section 7.2.2.
tcpIpNdpPrefix ListSize	PositiveInteger	0..1	attr	Maximum number of entries in the on-link prefix list.
tcpIpNdp Random ReachableTime Enabled	Boolean	0..1	attr	If enabled the value of ReachableTime will be multiplied with a random value between MIN_RANDOM_FACTOR and MAX_RANDOM_FACTOR in order to prevent multiple nodes from transmitting at exactly the same time.
tcpIpNdpRndRtr Solicitation DelayEnabled	Boolean	0..1	attr	If enabled the first router solicitation will be delayed randomly from [0...MAX_RTR_SOLICITATION_DELAY]. Otherwise the first router solicitation will be sent after exactly MAX_RTR_SOLICITATION_DELAY milliseconds.
tcpIpNdpRtr Solicitation Interval	TimeValue	0..1	attr	Interval between consecutive Router Solicitations in (s).
tcpIpNdpSlaac DadNumberOf Transmissions	PositiveInteger	0..1	attr	Number of Neighbor Solicitations that have to be unanswered in order to set an autoconfigurated address to PREFERRED (usable) state.
tcpIpNdpSlaac Dad Retransmission Delay	TimeValue	0..1	attr	Sets the maximum value for the address configuration delay (s).
tcpIpNdpSlaac DelayEnabled	Boolean	0..1	attr	If enabled transmission of the first DAD Neighbor Solicitation will be delayed by a random value from [0...MAX_DAD_DELAY].
tcpIpNdpSlaac OptimisticDad Enabled	Boolean	0..1	attr	Enable Optimistic Duplicate Address Detection (DAD) according to RFC4429.

Table 5.20: Ipv6NdpProps

5.2.2.2 TCP and UDP configuration properties

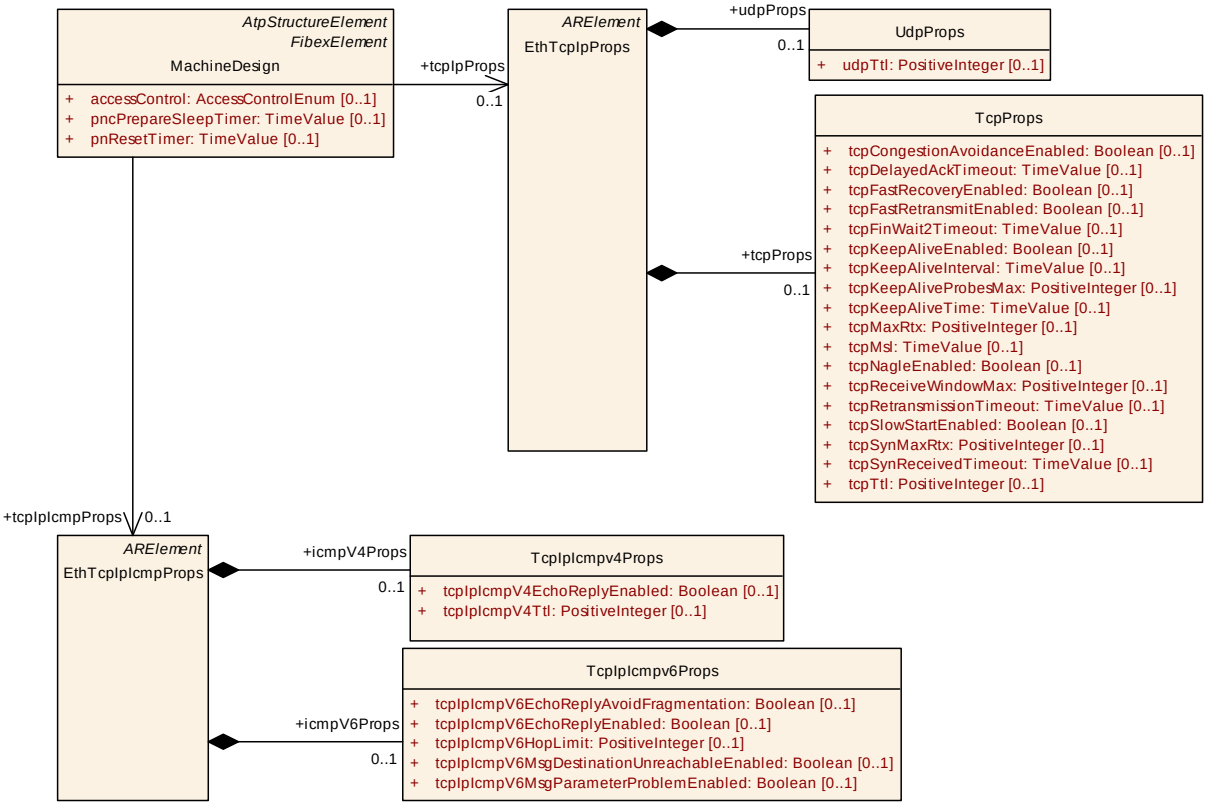


Figure 5.4: Machine specific TCP/UDP and ICMP configuration options

Class	**EthTcpIpProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class is used to configure the Machine specific TcpIp Stack attributes. **Tags:**atp.recommendedPackage=EthTcpIpProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpProps	TcpProps	0..1	aggr	TCP configuration properties
udpProps	UdpProps	0..1	aggr	UDP configuration properties

Table 5.21: EthTcpIpProps

Class	**UdpProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class specifies the configuration options for UDP (User Datagram Protocol).			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
udpTtl	PositiveInteger	0..1	attr	Default Time-to-live value of outgoing UDP packets.

Table 5.22: UdpProps

Class	**TcpProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class specifies the configuration options for TCP (Transmission Control Protocol).			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpCongestion Avoidance Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of TCP congestion avoidance algorithm according to IETF RFC 5681.
tcpDelayedAck Timeout	TimeValue	0..1	attr	The maximal time an acknowledgment is delayed for transmission in seconds.
tcpFast Recovery Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of TCP Fast Recovery according to IETF RFC 5681.
tcpFast Retransmit Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of TCP Fast Retransmission according to IETF RFC 5681.
tcpFin Wait2Timeout	TimeValue	0..1	attr	Timeout in [s] to receive a FIN from the remote node (after this node has initiated connection termination), i.e. maximum time waiting in FINWAIT-2 for a connection termination request from the remote TCP.
tcpKeepAlive Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) TCP Keep Alive Probes according to IETF RFC 1122 chapter 4.2.3.6.
tcpKeepAlive Interval	TimeValue	0..1	attr	Specifies the interval in seconds between subsequent keepalive probes.
tcpKeepAlive ProbesMax	PositiveInteger	0..1	attr	Maximum number of times that a TCP Keep Alive is retransmitted before the connection is closed.
tcpKeepAlive Time	TimeValue	0..1	attr	Specifies the time in [s] between the last data packet sent (simple ACKs are not considered data) and the first keepalive probe.
tcpMaxRtx	PositiveInteger	0..1	attr	Maximum number of times that a TCP segment is retransmitted before the TCP connection is closed. This parameter is only valid if tcpRetransmissionTimeout is configured. Note: This parameter also applies for FIN retransmissions.
tcpMsl	TimeValue	0..1	attr	Maximum segment lifetime in [s].
tcpNagle Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of Nagle's algorithm according to IETF RFC 1122 (chapter 4.2.3.4 When to Send Data). If enabled the Nagle's algorithm is activated per default for all TCP sockets, but can be deactivated per Socket (with the attribute TcpTp.nagle Algorithm).
tcpReceive WindowMax	PositiveInteger	0..1	attr	Default value of maximum receive window in bytes.
tcp Retransmission Timeout	TimeValue	0..1	attr	Timeout in [s] before an unacknowledged TCP segment is sent again. If the timeout is disabled, no TCP segments shall be retransmitted.
tcpSlowStart Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of TCP slow start algorithm according to IETF RFC 5681.
tcpSynMaxRtx	PositiveInteger	0..1	attr	Maximum number of times that a TCP SYN is retransmitted.
tcpSynReceived Timeout	TimeValue	0..1	attr	Timeout in [s] to complete a remotely initiated TCP connection establishment, i.e. maximum time waiting in SYN-RECEIVED for a confirming connection request acknowledgment after having both received and sent a connection request.

△

Class	TcpProps			
tcpTtl	PositiveInteger	0..1	attr	Default Time-to-live value of outgoing TCP packets.

Table 5.23: TcpProps

5.2.2.3 ICMP configuration properties

Class	EthTcpIpIcmpProps			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class is used to configure the Machine specific ICMP (Internet Control Message Protocol) attributes **Tags:**atp.recommendedPackage=EthTcpIcmpProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
icmpV4Props	TcpIpIcmpv4Props	0..1	aggr	ICMPv4 configuration properties
icmpV6Props	TcpIpIcmpv6Props	0..1	aggr	ICMPv6 configuration properties

Table 5.24: EthTcpIpIcmpProps

Class	TcpIpIcmpv4Props			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class specifies the configuration options for ICMPv4 (Internet Control Message Protocol).			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
tcpIpIcmp V4EchoReply Enabled	Boolean	0..1	attr	This attribute enables or disables transmission of ICMP echo reply message in case of a ICMP echo reception.
tcpIpIcmpV4Ttl	PositiveInteger	0..1	attr	This attribute is only relevant in case that ICMP (Internet Control Message Protocol) is used. It specifies the default Time-to-live value of outgoing ICMP packets.

Table 5.25: TcpIpIcmpv4Props

Class	TcpIpIcmpv6Props			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class specifies the configuration options for ICMPv6 (Internet Control Message Protocol).			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
tcpIpIcmp V6EchoReply Avoid Fragmentation	Boolean	0..1	attr	This attribute defines whether the echo reply is only transmitted in case that the incoming ICMPv6 Echo Request (Pings) fits the MTU of the respective interface, i.e. can be transmitted without IPv6 fragmentation.
tcpIpIcmp V6EchoReply Enabled	Boolean	0..1	attr	This attribute enables or disables transmission of ICMP echo reply message in case of a ICMP echo reception.
tcpIpIcmp V6HopLimit	PositiveInteger	0..1	attr	Default Hop-Limit value of outgoing ICMPv6 packets.

▽

△

Class	**TcpIpIcmpv6Props**			
tcpIpIcmp V6Msg Destination Unreachable Enabled	Boolean	0..1	attr	This attribute Enables/Disables the transmission of Destination Unreachable Messages.
tcpIpIcmp V6Msg Parameter Problem Enabled	Boolean	0..1	attr	If enabled an ICMPv6 parameter problem message will be sent if a received packet has been dropped due to unknown options or headers that are found in the packet.

Table 5.26: TcpIpIcmpv6Props

5.2.3 Securing Communication with IPsec

IPsec is a protocol suite that provides cryptographic protection for IP datagrams in IPv4 and IPv6 network packets.

IPsec uses a security association to specify security properties that are shared between the communicating parties. The security association defines a relationship between two or more parties and determines which security services will be used to communicate securely. In other words the security association serves as a “contract” between the different devices.

A single security association protects data in one communication direction. Two security associations shall be present to secure traffic in both directions. Each security association can provide encryption, data integrity and data authentication.

In addition, the senders and receivers of IP datagrams can determine the required protection for an IP packet according to IPsec security policies. These are rules that define how datagrams are processed that are received by a device. For example, security policies are used to decide if a particular packet needs to be dropped or needs to be processed by IPsec.

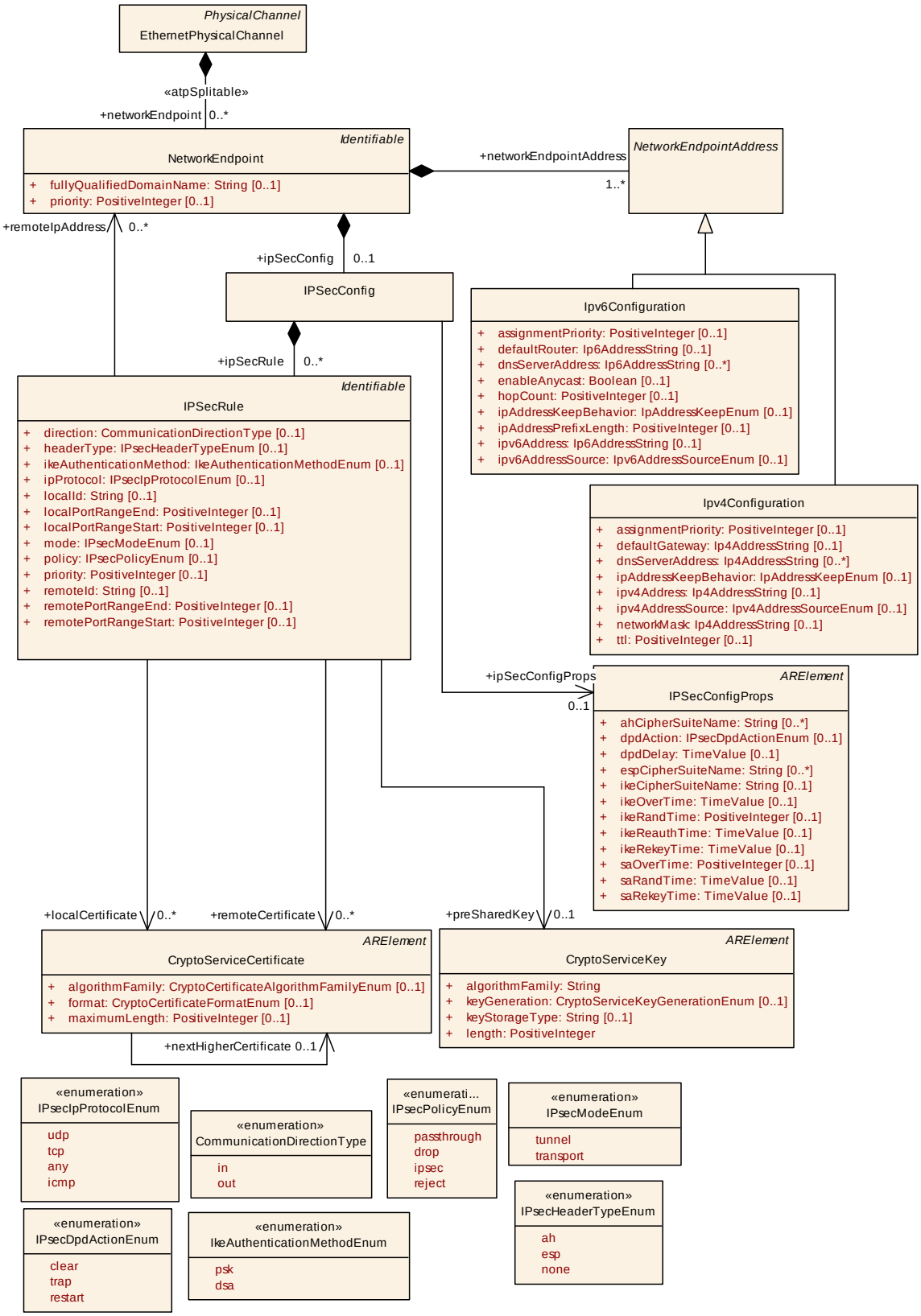


Figure 5.5: IPsec configuration model

[TPS_MANI_03203]{DRAFT} **Configuration of IPsec** ⌈The `IPSecConfig` meta-class that is aggregated by a `NetworkEndpoint` in the role `ipSecConfig` provides the ability to define IPsec settings that are necessary to configure IPsec security associations and IPsec security policies.⌋ *(RS_MANI_00036)*

[TPS_MANI_03204]{DRAFT} **Definition of `IPSecRules`** ⌈The `IPSecConfig` meta-class may contain one or several `IPSecRule`s. Each `IPSecRule` defines the network connection that is monitored by IPsec by defining the local endpoint and the remote endpoint. Each endpoint is defined by the IP Address and the Tcp/Udp Port. The communication direction for which the `IPSecRule` is valid is defined by the `direction` attribute.⌋ *(RS_MANI_00036)*

Class	**IPSecConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	IPsec is a protocol that is designed to provide "end-to-end" cryptographically-based security for IP network connections.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ipSecConfig Props	IPSecConfigProps	0..1	ref	Global IPsec configuration settings that are valid for all IPSecRules that are defined on the NetworkEndpoint.
ipSecRule	IPSecRule	*	aggr	IPSec rules and filters that are defined in the IPSecConfig for a specific NetworkEndpoint.

Table 5.27: IPSecConfig

Class	**IPSecRule**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This element defines an IPsec rule that describes communication traffic that is monitored, protected and filtered.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
direction	Communication DirectionType	0..1	attr	This attribute defines the direction in which the traffic is monitored. If this attribute is not set a bidirectional traffic monitoring is assumed.
headerType	IPsecHeaderTypeEnum	0..1	attr	Header type specifying the IPsec security mechanism.
ike Authentication Method	IkeAuthentication MethodEnum	0..1	attr	This attribute defines the IKE authentication method that is used locally and is expected on the remote side. **Tags:**atp.Status=obsolete
ipProtocol	IPsecIpProtocolEnum	0..1	attr	This attribute defines the relevant IP protocol used in the Security Policy Database (SPD) entry.
localCertificate	CryptoService Certificate	*	ref	This reference identifies the applicable certificate used for a local authentication. **Tags:**atp.Status=draft
localId	String	0..1	attr	This attribute defines how the local participant should be identified for authentication.

▽

△

Class	**IPSecRule**			
localPortRange End	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines an end value for the local port range. If this attribute is not set then this rule shall be effective for all local ports. Please note that port ranges are currently not supported in the AUTOSAR AP's operating system backend. If AP systems are involved, each IPsec rule may only contain a single port.
localPortRange Start	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines a start value for the local port range. If this attribute is not set then this rule shall be effective for all local ports. Please note that port ranges are currently not supported in the AUTOSAR AP's operating system backend. If AP systems are involved, each IPsec rule may only contain a single port.
mode	IPsecModeEnum	0..1	attr	This attribute defines the type of the connection.
policy	IPsecPolicyEnum	0..1	attr	An IPsec policy defines the rules that determine which type of IP traffic needs to be secured using IPsec and how that traffic is secured.
preSharedKey	CryptoServiceKey	0..1	ref	This reference identifies the applicable cryptograhic key used for authentication.
priority	PositiveInteger	0..1	attr	This attribute defines the priority of the IPSecRule (SPD entry). The processing of entries is based on priority, starting with the highest priority "0".
remote Certificate	CryptoService Certificate	*	ref	This reference identifies the applicable certificate used for a remote authentication. **Tags:**atp.Status=draft
remoteId	String	0..1	attr	This attribute defines how the remote participant should be identified for authentication.
remoteIp Address	NetworkEndpoint	*	ref	Definition of the remote NetworkEndpoint. With this reference the connection between the local Network Endpoint and the remote NetworkEndpoint is described on which the traffic is monitored.
remotePort RangeEnd	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines an end value for the remote port range. If this attribute is not set then this rule shall be effective for all local ports. Please note that port ranges are currently not supported in the AUTOSAR AP's operating system backend. If AP systems are involved, each IPsec rule may only contain a single port.
remotePort RangeStart	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines a start value for the remote port range. If this attribute is not set then this rule shall be effective for all local ports. Please note that port ranges are currently not supported in the AUTOSAR AP's operating system backend. If AP systems are involved, each IPsec rule may only contain a single port.

Table 5.28: IPSecRule

Class	**IPSecConfigProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This element holds all the attributes for configuration of IPsec that are independent of specific IPsec rules. **Tags:**atp.recommendedPackage=IPSecConfigProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ahCipherSuite Name	String	*	attr	AH (Authentication Header) algorithm to be used for the connection, e.g. HMAC/SHA2-256
dpdAction	IPsecDpdActionEnum	0..1	attr	This attribute defines what to do if the peer is considered dead. If not configured "restart" shall be assumed.
dpdDelay	TimeValue	0..1	attr	This attribute describes the interval to check the liveness of a peer actively using IKEv2 INFORMATIONAL exchanges. Active DPD checking is only enforced if no IKE or ESP/AH packet has been received for the configured DPD delay. In not configured the value "5 minutes" shall be assumed.
espCipherSuite Name	String	*	attr	ESP (Encapsulating Security Payload) algorithm that provides encryption and optional authentication for the connection, e.g. AES-128+SHA2-256.
ikeCipherSuite Name	String	0..1	attr	IKE encryption/authentication algorithms to be used for the connection.
ikeOverTime	TimeValue	0..1	attr	This attribute describes the hard deadline when an SA becomes invalid in percentage. Example: ikeOverTime of max(ikeReauthTime, ikeRekey Time). Default: 10 %
ikeRandTime	PositiveInteger	0..1	attr	This attribute defines in percentage by how long before the expiration of ikeReauthTime and ikeRekeyTime will be rekeyed/reauthenticated. Default: 10%
ikeReauthTime	TimeValue	0..1	attr	This attribute defines the absolute time after which an IKE SA will be reauthenticated. 0 means reauthentication is disabled.
ikeRekeyTime	TimeValue	0..1	attr	This attribute defines the absolute time after which an IKE SA will be rekeyed. 0 means rekey is disabled.
saOverTime	PositiveInteger	0..1	attr	This attribute describes the hard deadline when an IPsec SA becomes invalid in percentage. Example: saOverTime * saRekeyTime. Default: 110%
saRandTime	TimeValue	0..1	attr	This attribute defines by how long before the expiration of saRekeyTime will be rekeyed.
saRekeyTime	TimeValue	0..1	attr	This attribute defines the absolute time after which an IPsec SA will be rekeyed. 0 means rekey is disabled.

Table 5.29: IPSecConfigProps

Enumeration	**IPsecIpProtocolEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	Definition of supported TcpIp protocols that are supported in Security Policy Database (SPD) entries in IPSec configurations.
Literal	***Description***
any	ANY protocol **Tags:**atp.EnumerationLiteralIndex=3
icmp	Internet Control Message Protocol (ICMP) **Tags:**atp.EnumerationLiteralIndex=2
tcp	TCP Protocol **Tags:**atp.EnumerationLiteralIndex=1
udp	UDP Protocol **Tags:**atp.EnumerationLiteralIndex=0

Table 5.30: IPsecIpProtocolEnum

[constr_5102]{DRAFT} **Usage of remote port ranges in `IPSecRule` is not allowed** ⌈`IPSecRule.remotePortRangeStart` and `IPSecRule.remotePortRangeEnd` shall always be set to the same value.⌋*()*

[constr_5103]{DRAFT} **Usage of local port ranges in `IPSecRule` is not allowed** ⌈`IPSecRule.localPortRangeStart` and `IPSecRule.localPortRangeEnd` shall always be set to the same value.⌋*()*

The reason for [constr_5102] and [constr_5103] is that port ranges are currently not supported by the AUTOSAR Adaptive Platform operating system backend and each `IPSecRule` is allowed to define only a single local Port and a single remote Port.

[TPS_MANI_03232]{DRAFT} **Definition of general IPsec configuration settings** ⌈General configuration properties that are independent of particular `IPSecRule`s are collected in the `IPSecConfigProps` element that is referenced from the `IPSecConfig` in the role `ipSecConfigProps`.⌋*(RS_MANI_00036)*

[TPS_MANI_03205]{DRAFT} **IPsec policy** ⌈The `IPSecRule.policy` attribute defines how IP packets are handled that are going over the network connection defined by the `IPSecRule`. In detail, it defines whether the IP packet is processed by IPsec or not.⌋*(RS_MANI_00036)*

Enumeration	**IPsecPolicyEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	
Literal	***Description***
drop	Signifying that packets should be discarded **Tags:**atp.EnumerationLiteralIndex=3
ipsec	Signifying that packets should be protected. **Tags:**atp.EnumerationLiteralIndex=1

▽

△

Enumeration	**IPsecPolicyEnum**
passthrough	Signifying that no IPsec processing should be done at all. **Tags:**atp.EnumerationLiteralIndex=2
reject	Signifying that packets should be discarded and a diagnostic ICMP returned. **Tags:**atp.EnumerationLiteralIndex=4

Table 5.31: IPsecPolicyEnum

IPsec can be configured to operate in two different modes, Tunnel and Transport mode. With tunnel mode, the entire IP packet is protected by IPsec. IPsec wraps the original packet, encrypts it and adds a new IP header to it.

The tunnel mode is most commonly used between VPN gateways and the IP addresses of the newly added outer IP header are that of the VPN Gateways. In other words the traffic between the two VPN Gateways is protected and each gateway acts as a proxy for the hosts behind it.

The transport mode provides the protection of the Data Payload of the IP datagram with an AH or ESP header. The IP Header remains the same and IPsec inserts its header between the IP header and the upper level headers.

The IPsec transport mode can be used when securing traffic between two hosts or between a host and a VPN gateway.

[TPS_MANI_03233]{DRAFT} **IPsec mode** ⌈The `IPSecRule.mode` attribute defines whether the IP packet is processed in the `transport` or `tunnel` mode.⌋*(RS_MANI_00036)*

Please note that AUTOSAR currently supports only the `transport mode` as configuration option.

Enumeration	**IPsecModeEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This enumeration describes the supported IPSec modes.
Literal	***Description***
transport	Signifying that the IPSec transport mode is used. With the transport mode the original IP header is retained and only the IP payload and ESP trailer is encrypted. **Tags:**atp.EnumerationLiteralIndex=1
tunnel	Signifying that the IPSec tunnel mode is used. With tunnel mode, the entire original IP packet is protected by IPSec. This means IPSec wraps the original packet, encrypts it, adds a new IP header and sends it to the other side. **Tags:**atp.EnumerationLiteralIndex=0

Table 5.32: IPsecModeEnum

IPsec uses two protocols:

- AH - Authentication Header
- ESP - Encapsulating Security Payload

The AH protocol provides a mechanism for authentication only and authenticates the entire IP packet, including the outer IP header.

The ESP protocol provides data confidentiality (encryption) and/or authentication (data integrity, data origin authentication, and replay protection).

When ESP is used in transport mode, the IP payload is encrypted and the original IP header is moved to the front of the message. The ESP header is inserted after the IP header and is signed together with the IP payload. The original IP header remains unprotected.

When ESP is used in tunnel mode a new IP Header is created and the ESP header is added in front of the original IP Packet. The entire original IP packet is encrypted and signed in this mode.

[TPS_MANI_03206]{DRAFT} **IPsec AH and ESP protocol configuration** ⌈In the `IPSecRule` it is possible to define the IPsec protocol that shall be used to protect IP packets that are going over the defined network connection. The attribute `headerType` defines whether AH, ESP or neither one is used.⌋*(RS_MANI_00036)*

Enumeration	**IPsecHeaderTypeEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	IPsec Header Type options
Literal	***Description***
ah	Authentication Header (AH) **Tags:**atp.EnumerationLiteralIndex=0
esp	Encapsulating Security Payloads (ESP) **Tags:**atp.EnumerationLiteralIndex=1
none	No header **Tags:**atp.EnumerationLiteralIndex=2

Table 5.33: IPsecHeaderTypeEnum

[TPS_MANI_03234]{DRAFT} **IPsec AH and ESP CipherSuites** ⌈The attributes `ahCipherSuiteName` and `espCipherSuiteName` define the supported AH and ESP algorithms.⌋*()*

The naming convention for `ahCipherSuiteName`, `espCipherSuiteName` and `IPSecConfigProps.ikeCipherSuiteName` shall follow the naming convention for cryptographic primitives that is defined in [13].

[TPS_MANI_03207]{DRAFT} **IPsec Internet Key Exchange protocol configuration** ⌈In the `IPSecRule` it is possible to define how IKE protocol authenticates the remote party and how the local party authenticates itself to the remote party. In other words both sides use the same method. The usage of the `IPSecRule.preSharedKey` reference defines that the pre-shared key is used. The usage of the `IPSecRule.localCertificate` and `IPSecRule.remoteCertificate` defines that Digital Signature Authentication is used.⌋*()*

Enumeration	**IkeAuthenticationMethodEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This enumeration describes the supported IKE authentication methods. **Tags:**atp.Status=obsolete
Literal	***Description***
dsa	Digital Signature Authentication **Tags:**atp.EnumerationLiteralIndex=2
psk	Pre-shared key authentication **Tags:**atp.EnumerationLiteralIndex=1

Table 5.34: IkeAuthenticationMethodEnum

Please note that the supported IKE CipherSuites are configured with the `IPSecConfigProps.ikeCipherSuiteName`. The `IPSecConfigProps` contains additional IKE specific configuration settings.

Enumeration	**IPsecDpdActionEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	Potential Dead Peer Detection (Dpd) Actions
Literal	***Description***
clear	Deletes the SA. **Tags:**atp.EnumerationLiteralIndex=0
restart	Immediately tries to establish the connection. **Tags:**atp.EnumerationLiteralIndex=2
trap	tries to establish the connection after traffic is sent to the peer. **Tags:**atp.EnumerationLiteralIndex=1

Table 5.35: IPsecDpdActionEnum

[TPS_MANI_03208]{DRAFT} **Protection of `AdaptivePlatformServiceInstance` by IPsec** ⌈To describe the protection of an `AdaptivePlatformServiceInstance` by IPsec the `AdaptivePlatformServiceInstance` needs to be mapped by a `ServiceInstanceToMachineMapping` to an `EthernetCommunicationConnector` that points with the `unicastNetworkEndpoint` to a `NetworkEndpoint` that aggregates the `IPSecConfig` that in turn describes IPsec Security Associations.⌋*(RS_MANI_00036)*

Please note that IP Multicast protection by IPsec is not supported. It is by intention not possible to model the IPsec protection of IP Multicast communication since the IP Multicast address is defined in the `SomeipProvidedEventGroup` by the two attributes `ipv4MulticastIpAddress` and `ipv6MulticastIpAddress`. The `NetworkEndpoint` element is used for description of IP Unicast Endpoints only. This means that only the IP Unicast communication of an `AdaptivePlatformServiceInstance` that is described according to [TPS_MANI_03208] will be protected by IPsec.

5.2.4 Service Discovery Configuration

Service Discovery messages are exchanged between network nodes to announce and to discover available service instances. This chapter describes the configuration that is necessary to exchange service discovery messages for supported middleware transport layers.

Class	***ServiceDiscoveryConfiguration*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	Service Discovery configuration settings for the middleware transport layer. **Tags:**atp.Status=draft			
Base	*ARObject*			
Subclasses	SomeipServiceDiscovery			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 5.36: ServiceDiscoveryConfiguration

5.2.4.1 SOME/IP Service Discovery Configuration

[TPS_MANI_03064]{DRAFT} **SOME/IP Service Discovery message exchange configuration** ⌈ProvidedServiceInstances are announced in SOME/IP by the server with multicast addressing on a VLAN to a specifically designated IP multicast address (SomeipServiceDiscovery.multicastSdIpAddress) at a specific UDP port number (SomeipServiceDiscovery.someipServiceDiscoveryPort).⌋*(RS_MANI_00019)*

[constr_5045]{DRAFT} **Only one SomeipServiceDiscovery configuration per VLAN is allowed** ⌈Only a single NetworkEndpoint on an EthernetPhysicalChannel (VLAN) is allowed to be referenced by a SomeipServiceDiscovery element in the role multicastSdIpAddress.⌋*()*

The SomeipServiceDiscovery is able to reference SecureComProps to define and to configure a security protocol that will provide communication security for Service Discovery messages.

For Service Discovery messages that will be transmitted to a designated multicast IP address the protection is defined by the SecureComProps that is referenced in the role multicastSecureComProps. For unicast Service Discovery messages different credentials may be used for the different ECU pairs.

Therefore, a list of SecureComProps is aggregated in the role unicastSecureComProps.

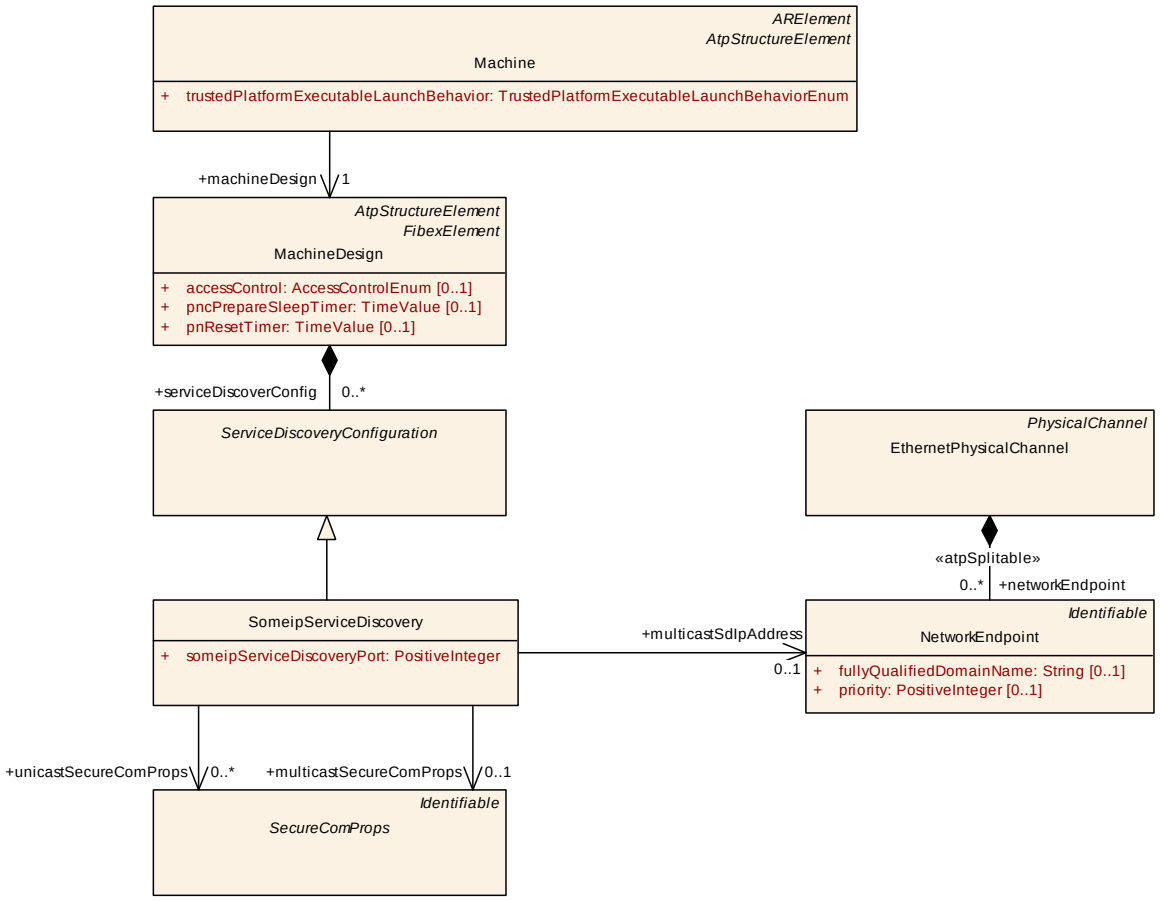


Figure 5.6: SOME/IP Service Discovery Configuration

Class	**SomeipServiceDiscovery**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	This meta-class represents a specialization of the generic service discovery for the SOME/IP case. **Tags:**atp.Status=draft			
Base	*ARObject*, *ServiceDiscoveryConfiguration*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
multicastSdIp Address	NetworkEndpoint	0..1	ref	This reference identifies the multicast IP address used for service discovery. **Tags:**atp.Status=draft
multicastSecure ComProps	SecureComProps	0..1	ref	Reference to a communication security protocol and its configuration settings that will provide communication security for Service Discovery messages that are transmitted using multicast, e.g. FindService message. **Tags:**atp.Status=draft
someipService DiscoveryPort	PositiveInteger	1	attr	This attribute represents the port number reserved for service discovery.

△

Class	**SomeipServiceDiscovery**			
unicastSecure ComProps	SecureComProps	*	ref	Reference to a communication security protocol and its configuration settings that will provide communication security for Service Discovery messages that are transmitted using unicast, e.g. OfferService as answer to a FindService message. . **Tags:**atp.Status=draft

Table 5.37: SomeipServiceDiscovery

5.2.5 Partial Network

AUTOSAR supports power saving during vehicle operation time with the partial networking mechanism. This mechanism allows shutting down and starting up the bus communication interfaces of groups of ECUs (Partial Network Cluster) during normal bus communication.

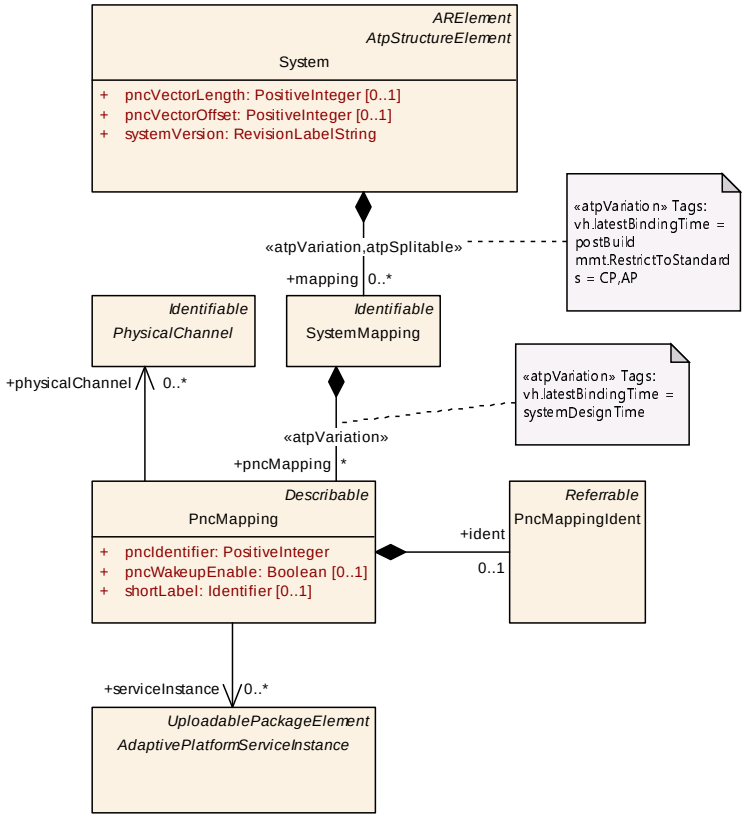


Figure 5.7: PncMapping with collection of ServiceInstances that are participating in the Partial Network Cluster

On the VFB Level Partial Networks are represented by Virtual Function Clusters and are described with PortGroups. The Virtual Function Cluster groups the communication necessary to realize one or more vehicle functions that can become activated/deactivated during normal vehicle operation. The Virtual Function Clusters are mapped onto Partial Network Clusters.

[TPS_MANI_03224]{DRAFT} **Modeling of a Partial Network Cluster** ⌈A Partial Network Cluster is modeled with the PncMapping element and is identified by the pncIdentifier. The PncMapping defines the collection of AdaptivePlatformServiceInstances that are participating in the partial network with the PncMapping.serviceInstance reference.⌋*(RS_MANI_00062)*

[TPS_MANI_03225]{DRAFT} **References to vlans in PncMapping** ⌈An EthernetCommunicationConnector may be referenced directly by a given PncMapping in the role physicalChannel and also by a ServiceInstanceToMachineMapping that maps an AdaptivePlatformServiceInstance to a EthernetCommunicationConnector that in turn is referenced by the same PncMapping in the role serviceInstance.⌋*(RS_MANI_00062)*

Class	**SystemMapping**			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	The system mapping aggregates all mapping aspects that are relevant in the System Description.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
pncMapping	PncMapping	*	aggr	Mappings between Virtual Function Clusters and Partial Network Clusters. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=systemDesignTime

Table 5.38: SystemMapping

Class	**PncMapping**			
Package	M2::AUTOSARTemplates::SystemTemplate::PncMapping			
Note	Describes a mapping between one or several Virtual Function Clusters onto Partial Network Clusters. A Virtual Function Cluster is realized by a PortGroup. A Partial Network Cluster is realized by one or more ServiceInstances.			
Base	*ARObject*, *Describable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
ident	PncMappingIdent	0..1	aggr	This adds the ability to become referrable to PncMapping.
physical Channel	PhysicalChannel	*	ref	This reference maps the partial network to a communication channel.
pncConsumed ProvidedServiceInstance Group	ConsumedProvided ServiceInstanceGroup	*	ref	ConsumedProvidedServiceInstanceGroup used in a Partial Network Cluster. This reference is optional, since this could be used for starting and stopping Consumed ProvidedServiceInstanceGroup according the requested partial network, but is not necessarily needed. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild

▽

△

Class	**PncMapping**			
pncIdentifier	PositiveInteger	1	attr	Identifer of the Partial Network Cluster. This number represents the absolute bit position of this Partial Network Cluster in the NM Pdu.
pncWakeup Enable	Boolean	0..1	attr	If this parameter is available and set to true then this PNC will be woken up as soon as a channel wakeup occurs on a channel where this PNC is assigned to. This is ensured by adding this PNC to the corresponding channel wakeup sources during upstream mapping.
relevantFor DynamicPnc Mapping	EcuInstance	*	ref	Reference to a PNC Gateway ECU for PNCs which do not have a static channel mapping. This is needed to describe dynamic PNCs that can be learned only at run-time and which have no relation to an ISignalIPdu Group. **Tags:**atp.Status=draft
serviceInstance	AdaptivePlatform ServiceInstance	*	ref	Reference to ServiceInstances that are participating in a Partial Network Cluster. **Tags:**atp.Status=draft
shortLabel	Identifier	0..1	attr	This attribute specifies an identifying shortName for the PncMapping. It shall be unique in the System scope.
vfc	PortGroup	*	iref	Virtual Function Cluster to be mapped onto a Partial Network Cluster. This reference is optional in case that the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy systems. **InstanceRef implemented by:**PortGroupInSystem InstanceRef

Table 5.39: PncMapping

5.3 Log and Trace Design

The Log and Trace functionality in AUTOSAR supports the monitoring of applications and provides means to forward logging information onto the communication bus. The log channel defines the log and trace message output for a source that is monitored.

The modeling of such log channels is done on two levels:

- the definition of `DltLogChannelDesign` allows for the pre-specification of log channels already on the design level. The modeling of `DltLogChannelDesign` is described in this chapter.
- the definition of `DltLogChannel` allows for the actual and final specification of log channels from the perspective of the platform software. The modeling of `DltLogChannel` is described in chapter 9.6.

[TPS_MANI_03275]{DRAFT} **Configuration of log and trace message source on design level** ⌈The `DltLogChannelDesign` is used to to configure the log and trace message for a source defined by the application that the log and trace message originates. The relationship to the `Executable` is created by the `DltLogChannelDesignToProcessDesignMapping` that maps the `DltLogChannelDesign` to a `ProcessDesign` that in turn refers the `Executable` in the `executable` role.⌋*()*

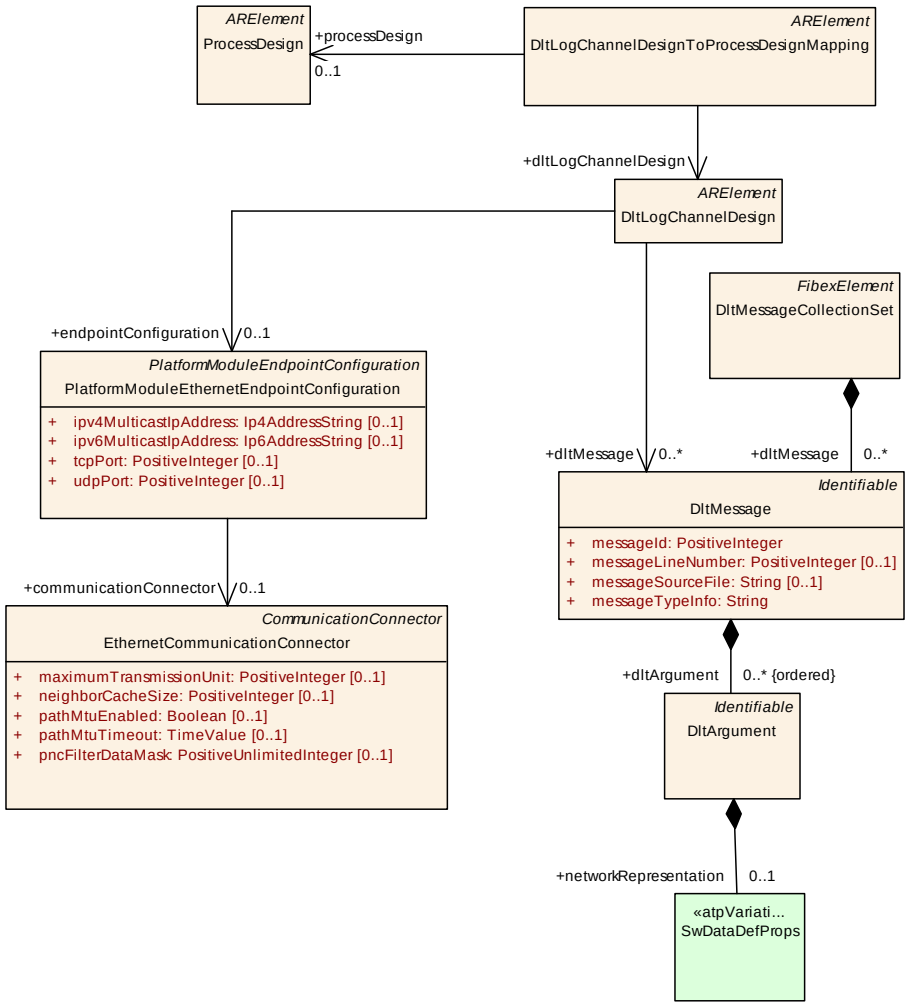


Figure 5.8: Log and Trace Design modeling

Class	**DltLogChannelDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This meta-class has the ability to stand in for a DltLogChannel at the time when the DltLogChannel does not yet exist. But its future existence already needs to be considered during design phase and for that a dedicated model element is required. **Tags:** atp.Status=draft atp.recommendedPackage=DltLogChannelDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dltMessage	DltMessage	*	ref	Reference to DltMessages that can be transported over the DltLogChannel. **Tags:**atp.Status=draft
endpoint Configuration	PlatformModule EthernetEndpoint Configuration	0..1	ref	Network configuration (Protocol, Port, IP Address) for transmission of dlt messages on a specific VLAN. **Tags:**atp.Status=draft

Table 5.40: DltLogChannelDesign

Class	**DltLogChannelDesignToProcessDesignMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This meta-class represents the ability to assign a Log&Trace Channel in the Design to a ProcessDesign. **Tags:** atp.Status=draft atp.recommendedPackage=DltLogChannelDesignToProcessDesignMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dltLogChannel Design	DltLogChannelDesign	1	ref	Reference to the Log&Trace channel that contains the log/trace message output. **Tags:**atp.Status=draft
processDesign	ProcessDesign	0..1	ref	Reference to the ProcessDesign that is monitored by the DltLogChannel. **Tags:**atp.Status=draft

Table 5.41: DltLogChannelDesignToProcessDesignMapping

5.4 Specification of Application Software System Structure

The root element of a System Design model is the `System` element that is already known from the AUTOSAR classic platform. The `System` aggregates the `RootSwCompositionPrototype` that represents the top-level-composition of all software components that are available in a given system.

[TPS_MANI_03110]{DRAFT} **Allowed components in system description with `category SYSTEM_DESIGN_DESCRIPTION`.** ⌈`SwComponentPrototype`s nested inside the `CompositionSwComponentType` that is referenced by the `RootSwCompositionPrototype` of a `System` with `category SYSTEM_DESIGN_DESCRIPTION` are allowed to be of any `SwComponentType` that is supported by Classic or by Adaptive Autosar.⌋*(RS_MANI_00026)*

Class	**RootSwCompositionPrototype**
Package	M2::AUTOSARTemplates::SystemTemplate
Note	The RootSwCompositionPrototype represents the top-level-composition of software components within a given System. According to the use case of the System, this may for example be a more or less complete VFB description, the software of a System Extract or the software of a flat ECU Extract with only atomic SWCs. Therefore the RootSwComposition will only occasionally contain all atomic software components that are used in a complete VFB System. The OEM is primarily interested in the required functionality and the interfaces defining the integration of the Software Component into the System. The internal structure of such a component contains often substantial intellectual property of a supplier. Therefore a top-level software composition will often contain empty compositions which represent subsystems. The contained SwComponentPrototypes are fully specified by their SwComponentTypes (including Port Prototypes, PortInterfaces, VariableDataPrototypes, SwcInternalBehavior etc.), and their ports are interconnected using SwConnectorPrototypes.

▽

△

Class	**RootSwCompositionPrototype**			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
software Composition	CompositionSw ComponentType	1	tref	We assume that there is exactly one top-level composition that includes all Component instances of the system **Stereotypes:** isOfType

Table 5.42: RootSwCompositionPrototype

If a Software Component communicates over the service oriented communication and provides or requires a `ServiceInterface` the opposite communication end is not always known upfront. In the `System` with `category SYSTEM_DESIGN_DESCRIPTION` a System Designer may want to indicate the service oriented communication between endpoints if it is already known at the System Design time.

[TPS_MANI_03114]{DRAFT} **Usage of `AssemblySwConnector`s in the System Design model** ⌈In the `System` with `category SYSTEM_DESIGN_DESCRIPTION` it is allowed to indicate the service oriented communication between two communication endpoints by `AssemblySwConnector`s if the required `RPortPrototype` is searching for a specific service instance, i.e. if the `RPortPrototypeProps.searchIntention` is set to `searchForId`.

If the `searchIntention` is set to `searchForAll`, the `AssemblySwConnector` shall not be used to connect this `RPortPrototype`.⌋*(RS_MANI_00026)*

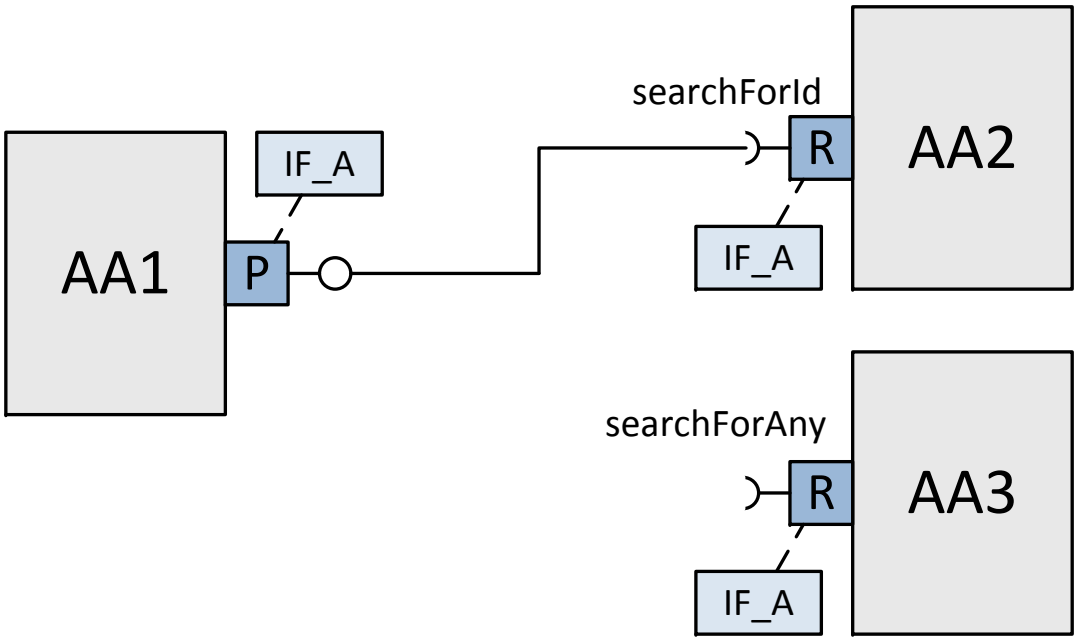


Figure 5.9: Example for Assembly connectors in System Design model

5.5 Modeling of service oriented communication between Classic and Adaptive platform

AUTOSAR classic platform does not support ServiceInterfaces yet but provides the possibility to communicate in a service oriented way over SOME/IP. To mimic a ServiceInterface in the classic platform any combination of ClientServerInterfaces, SenderReceiverInterfaces or TriggerInterfaces may be used to describe a service to which later a SOME/IP Service ID is assigned.

To simplify the description of the service oriented communication between Classic and Adaptive Software components in a System design model the InterfaceMapping was introduced that allows to map elements of PortInterfaces of the Classic Platform to a single ServiceInterface of the Adaptive Platform.

Class	**InterfaceMappingSet**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This meta-class represents the ability to aggregate a collection of InterfaceMappings. **Tags:** atp.Status=draft atp.recommendedPackage=InterfaceMappingSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
interfaceMapping	InterfaceMapping	*	aggr	Mapping of a ServiceInterface of the Adaptive Platform to PortInterface elements of the Classic Platform. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=interfaceMapping.shortName, interfaceMapping.variationPoint.shortLabel atp.Status=draft vh.latestBindingTime=systemDesignTime

Table 5.43: InterfaceMappingSet

Class	**InterfaceMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This meta-class collects the mappings of elements of a single ServiceInterface to PortInterface elements of the AUTOSAR Classic Platform. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventMapping	EventMapping	*	aggr	Mapping of a VariableDataPrototype in a SenderReceiver Interface to an Event in a ServiceInterface. **Tags:**atp.Status=draft
fieldMapping	FieldMapping	*	aggr	Mapping of a Field in a ServiceInterface to ClientServer Operations that represent the getter and setter methods and to a VariableDataPrototype that represents the notifier in the Field. **Tags:**atp.Status=draft

△

Class	**InterfaceMapping**			
fireAndForget Mapping	FireAndForgetMapping	*	aggr	Mapping of a Fire&Forget Method that is located in a ServiceInterface to a VariableDataPrototype in a Sender ReceiverInterface or to a Trigger in a TriggerInterface. **Tags:**atp.Status=draft
methodMapping	MethodMapping	*	aggr	Mapping of a ClientServerOperation in a ClientServer Interface to a Method in a ServiceInterface. **Tags:**atp.Status=draft

Table 5.44: InterfaceMapping

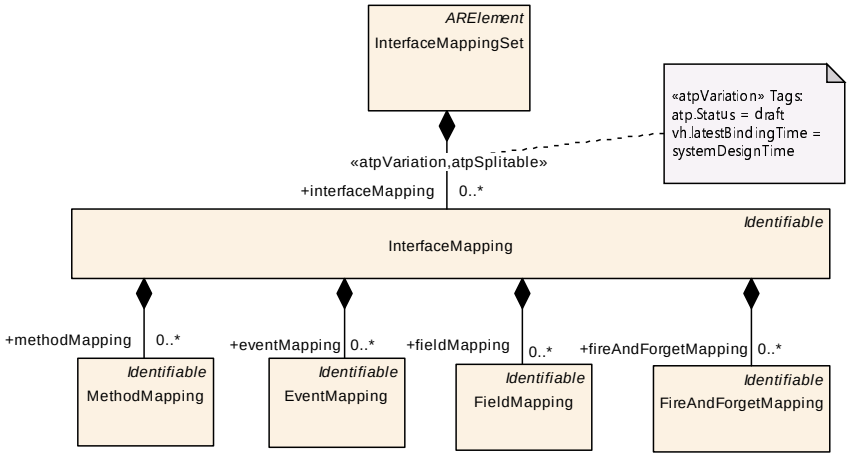


Figure 5.10: InterfaceMapping Overview

[constr_3370]{DRAFT} **`InterfaceMapping` shall map all elements of a single `ServiceInterface`** ⌈The mappings that are included in an `InterfaceMapping` shall map all elements of a single `ServiceInterface` (i.e. `field`s, `event`s, `method`s) to `PortInterface` elements of the classic platform.⌋*()*

Figure 5.11 shows a possible System Design modeling approach where an Adaptive Application is communicating in a service oriented way over SOME/IP with classic Software Components. SWC_1 requires a `ClientServerInterface` IF_Y with a `ClientServerOperation` and a `SenderReceiverInterface` IF_X with a `VariableDataPrototype`. SWC_2 requires a `SenderReceiverInterface` IF_X with a `VariableDataPrototype`.

The two `PortInterface`s IF_X and IF_Y are mapped to a single `ServiceInterface` IF_A using an `InterfaceMapping`.

On the other side the Adaptive Application AA1 provides the `ServiceInterface` IF_A.

Note that this is a mapping on `PortInterface` level. If each `PortInterface` is only used once in a network the actual communication can be directly derived out of the `InterfaceMapping`. If `PortInterface`s are used several times on a network there is the need to take the network configuration into account in order to be able to emulate

how the service discovery will behave on the network. From this information the actual communication relations on software level can be deduced.

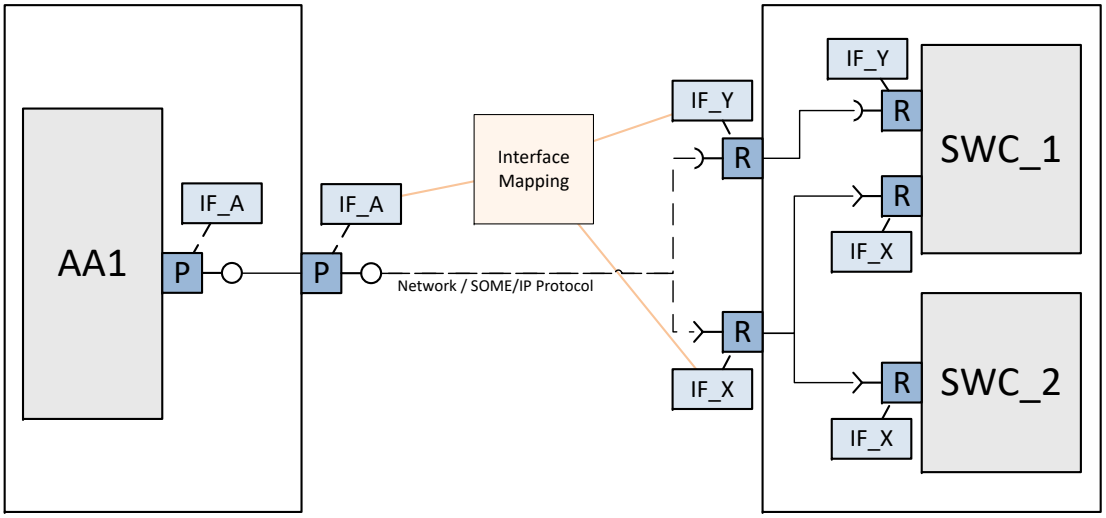


Figure 5.11: Example for a modeling of Service Oriented communication between Adaptive Applications and Software Components of the Classic Platform

5.5.1 MethodMapping

[TPS_MANI_03111]{DRAFT} **Mapping between `method` and `operation`located in a `ClientServerInterface`** ⌈The mapping between a `method` located in a `ServiceInterface` and a `operation` located in a `ClientServerInterface` is provided by the class `MethodMapping`.⌋*(RS_MANI_00026)*

Class	**MethodMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	Mapping of a ClientServerOperation that is located in a ClientServerInterface to a Method that is located in a ServiceInterface. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientServer Operation	ClientServerOperation	0..1	ref	Reference to a ClientSeverOperation that is located in a ClientSeverInterface. **Tags:**atp.Status=draft
method	ClientServerOperation	0..1	ref	Reference to a Method that is located in a Service Interface. **Tags:**atp.Status=draft

Table 5.45: MethodMapping

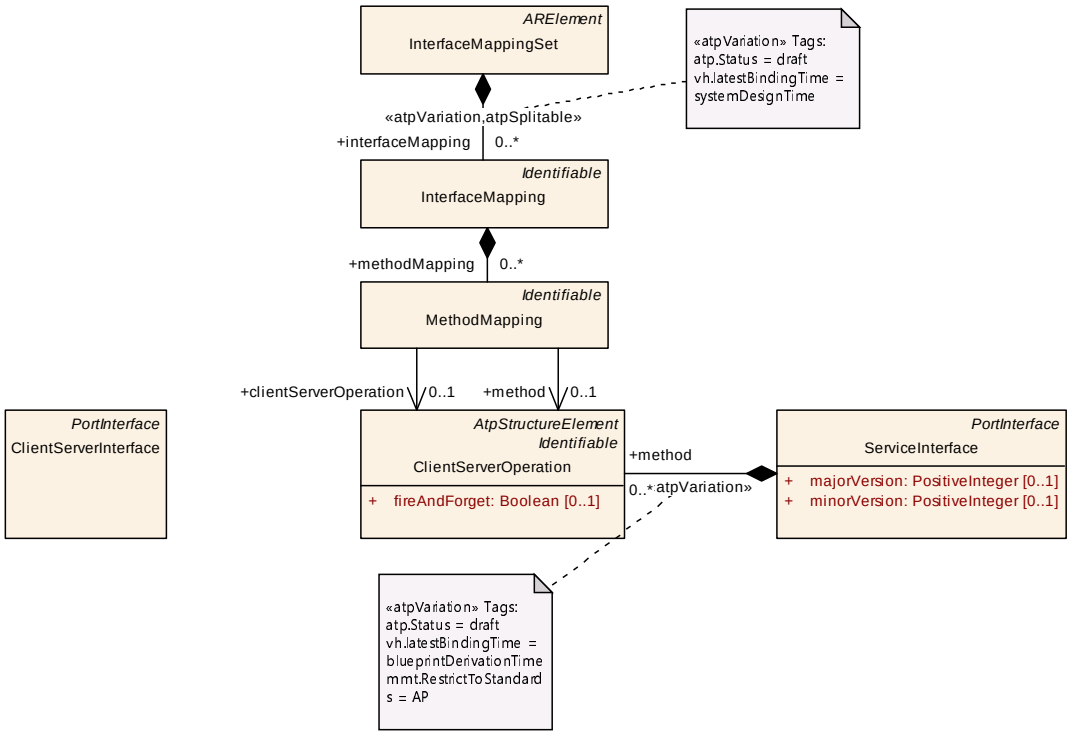


Figure 5.12: Mapping of a Method to a ClientServerOperation

5.5.2 EventMapping

[TPS_MANI_03112]{DRAFT} **Mapping between an `event` and a `dataElement`**
⌈The mapping between an `event` located in a `ServiceInterface` and a `dataElement` located in a `SenderReceiverInterface` is provided by the class `EventMapping`.⌋*(RS_MANI_00026)*

Class	**EventMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	Mapping of a VariableDataPrototype that is located in a SenderReceiverInterface to an Event that is located in a ServiceInterface. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	VariableDataPrototype	0..1	ref	Reference to a VariableDataPrototype that is located in a SenderReceiverInterface. **Tags:**atp.Status=draft
event	VariableDataPrototype	0..1	ref	Reference to an Event that is located in a Service Interface. **Tags:**atp.Status=draft

Table 5.46: EventMapping

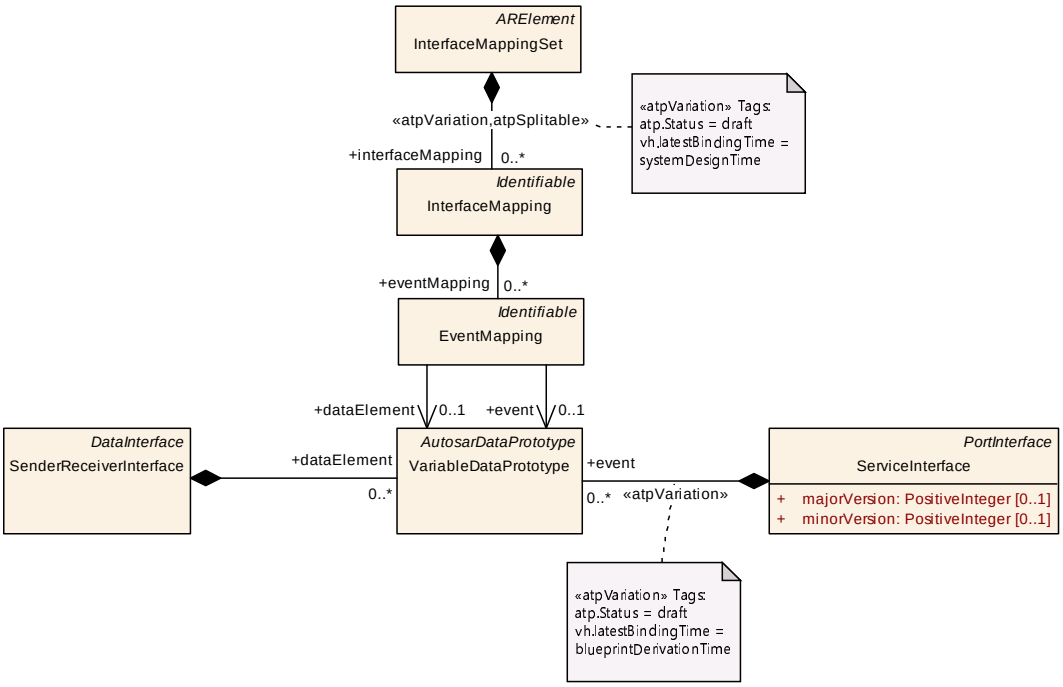


Figure 5.13: Mapping between an `event` and a `dataElement`

5.5.3 FieldMapping

[TPS_MANI_03113]{DRAFT} **Mapping between a `field` and elements of Classic Platform `PortInterface`s** ⌈The mapping between a `field` located in a `ServiceInterface` and elements of Classic Platform `PortInterface`s is provided by the class `FieldMapping`. The field notifier in the classic platform is represented by a `dataElement` that is located in a `SenderReceiverInterface`. The getter and setter methods in the classic platform are represented by `operation`s that are located in a `ClientServerInterface`.⌋*(RS_MANI_00026)*

[constr_3367]{DRAFT} **`FieldMapping.notifierDataElement` reference** ⌈The `FieldMapping` shall only contain the `notifierDataElement` reference if the `hasNotifier` attribute in the referenced `field` is set to true.⌋*()*

[constr_3368]{DRAFT} **`FieldMapping.getterOperation` reference** ⌈The `FieldMapping` shall only contain the `getterOperation` reference if the `hasGetter` attribute in the referenced `field` is set to true.⌋*()*

[constr_3369]{DRAFT} **`FieldMapping.setterOperation` reference** ⌈The `FieldMapping` shall only contain the `setterOperation` reference if the `hasSetter` attribute in the referenced `field` is set to true.⌋*()*

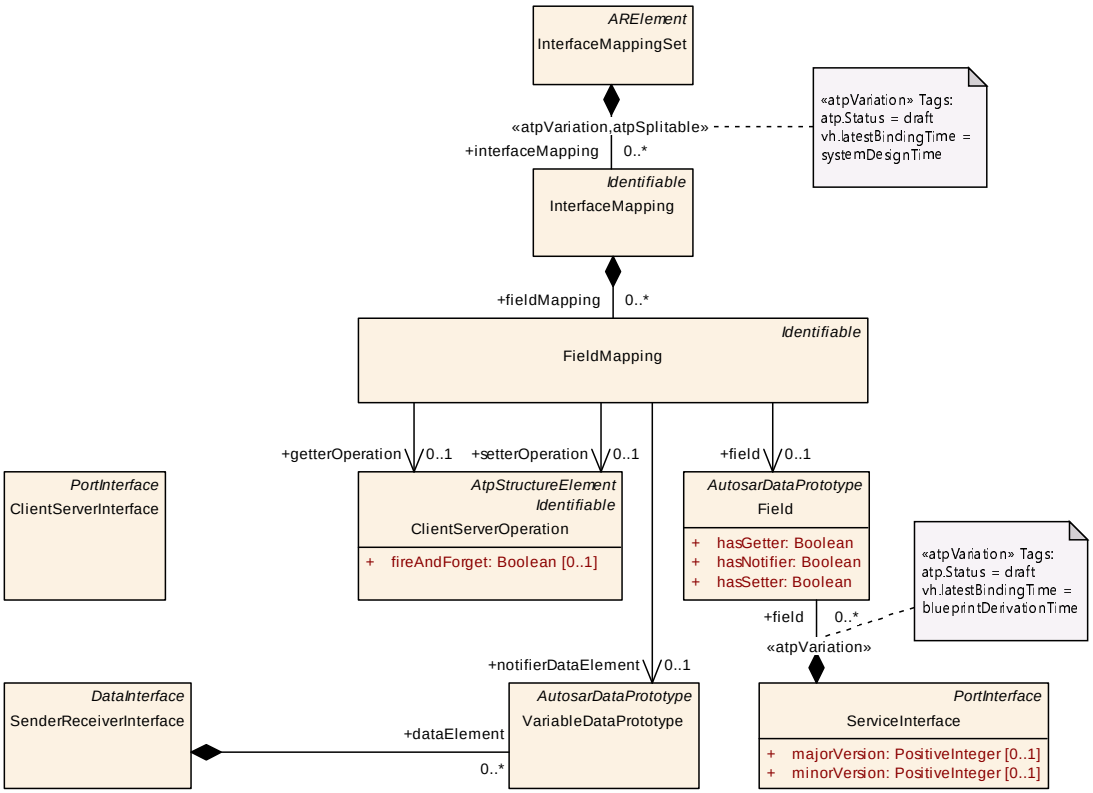


Figure 5.14: Mapping between a `field` and elements of Classic Platform `PortInterfaces`

Class	**FieldMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	Mapping of a Field that is located in a ServiceInterface to ClientServerOperations that represent the getter and setter methods and to a VariableDataPrototype that represents the notifier in the Field. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
field	Field	0..1	ref	Reference to a field that is located in a ServiceInterface. **Tags:**atp.Status=draft
getterOperation	ClientServerOperation	0..1	ref	Reference to a ClientServerOperation that represents the getter Method in the Field. **Tags:**atp.Status=draft
notifierData Element	VariableDataPrototype	0..1	ref	Reference to a VariableDataPrototype that represents the notifier in the Field. **Tags:**atp.Status=draft
setterOperation	ClientServerOperation	0..1	ref	Reference to a ClientServerOperation that represents the setter Method in the Field. **Tags:**atp.Status=draft

Table 5.47: FieldMapping

5.5.4 FireAndForgetMapping

In a fire and forget Message Exchange Pattern the consumer sends a message to a provider with no expectation of a response as described in chapter 3.4.4.1.

In Adaptive Autosar the fire and forget method is described with a `method` where the value of attribute `method.fireAndForget` is set to `true` as defined by [TPS_MANI_01064].

In classic Autosar a fire and forget method can not be described with a `ClientServerOperation` since a client-server call always has a response. Therefore, a `VariableDataPrototype` is used if the fire and forget method contains input arguments.

If the fire and forget method contains several input arguments then the `VariableDataPrototype` needs to be of type Structure that hosts one element for each argument of the fire and forget method. It is important that the order of elements in the Structure is the same as the order of `ArgumentDataPrototype`s within the `ClientServerOperation`.

This representation ensures that the SOME/IP serialization results in the same byte stream as in the Adaptive Platform where all `argument`s which have the `direction in` are serialized according to the order of the `ArgumentDataPrototype`s within the `ClientServerOperation`.

If the fire and forget method is without any parameters a `Trigger` is used to describe such a method in classic Autosar.

It is important that the SOME/IP MessageType is set to REQUEST_NO_RETURN if a fire and forget method is transmitted over SOME/IP.

[TPS_MANI_03115]{DRAFT} **Mapping between a fire and forget `method` and elements of Classic Platform `PortInterface`s** ⌈The mapping between a `method` for which the value of attribute `method.fireAndForget` is set to `true` and elements of Classic Platform `PortInterface`s is provided by the class `FireAndForgetMapping`.

If the fire and forget method is represented in the classic platform by a `VariableDataPrototype` then this `dataElement` is mapped to a `method` located in a `ServiceInterface`. If the fire and forget method is represented in the classic platform by a `Trigger` then this `trigger` is mapped to a `method` located in a `ServiceInterface`.⌋*(RS_MANI_00026)*

[constr_3371]{DRAFT} **Mutually exclusive existence of `FireAndForgetMapping.dataElement` reference and `FireAndForgetMapping.trigger` reference** ⌈A `FireAndForgetMapping` shall never reference a `dataElement` and a `trigger` at the same time.⌋*()*

[constr_3376]{DRAFT} **`FireAndForgetMapping` shall reference only fire and forget `method`s** ⌈A `FireAndForgetMapping` is only allowed to reference a

ClientServerOperation in role method for which the value of attribute method.fireAndForget is set to true.⌋()

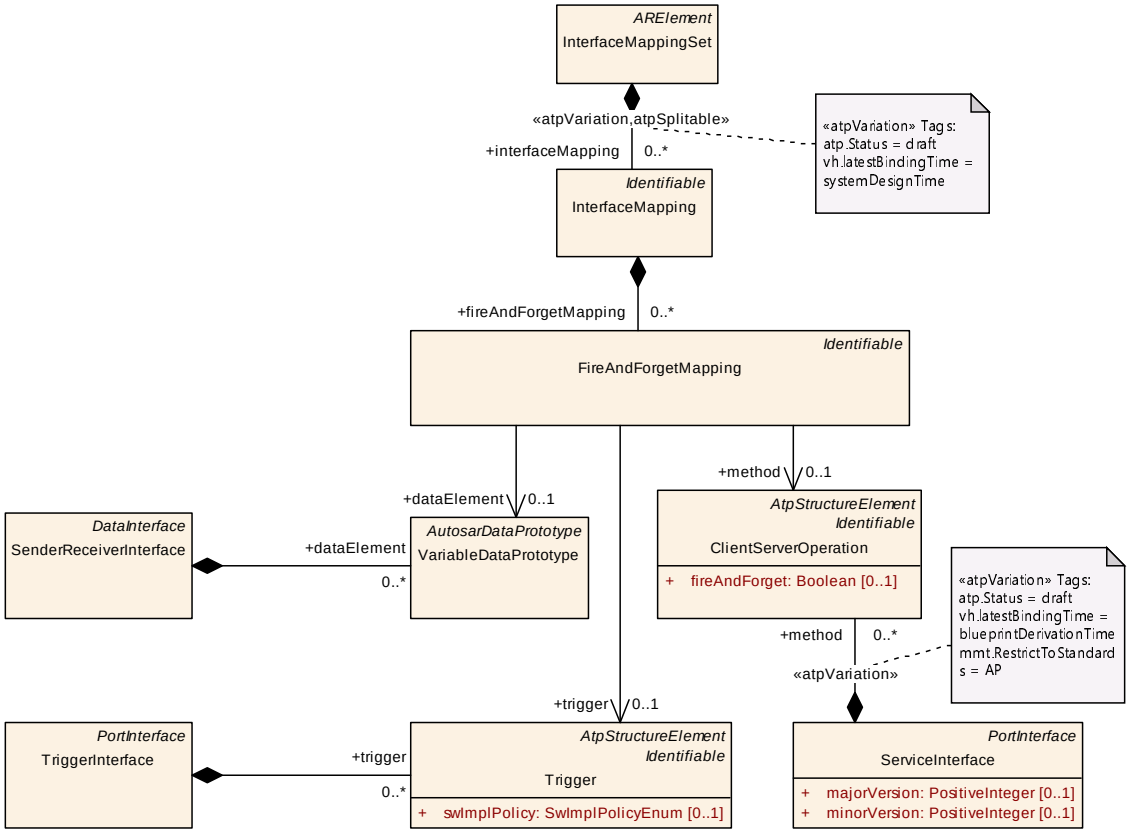


Figure 5.15: Mapping between a fire and forget method and elements of Classic Platform PortInterfaces

Class	**FireAndForgetMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	Mapping of a Fire&Forget Method that is located in a ServiceInterface to a VariableDataPrototype in a SenderReceiverInterface or to a Trigger in a TriggerInterface. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	VariableDataPrototype	0..1	ref	Reference to a VariableDataPrototype that is located in a SenderReceiverInterface in case that the Fire&Forget Method is represented by this VariableDataPrototype. **Tags:**atp.Status=draft
method	ClientServerOperation	0..1	ref	Reference to a Fire&Forget Method that is located in a ServiceInterface. **Tags:**atp.Status=draft

▽

△

Class	**FireAndForgetMapping**			
trigger	Trigger	0..1	ref	Reference to a Trigger that is located in a TriggerInterface in case that the Fire&Forget Method is represented by this Trigger. **Tags:**atp.Status=draft

Table 5.48: FireAndForgetMapping

6 Sub-System Design

6.1 Overview

The nature of the *AUTOSAR adaptive platform* as a platform for deploying software units in the field implies that the software units that can be installed in the field need some design-support upfront.

More specifically, the software units that can be deployed in the field typically represent some sort of more or less self-contained driving function.

In other words, the design support for this purpose need to be tailored to facilitate the design of application-level software that communicates with other application level software.

It is assumed that one of the first steps in such a design is the definition of services that are provided and services that are required by the driving function under development.

Such a definition of required and provided services can be used as an input into the design of other such driving functions and, over time, a view of the communication on the level of driving functions is rendered.

It is further assumed that the communication view of the driving functions is mostly of interest for an OEM and the individual driving functions may be sub-contracted to tier-1 suppliers.

This means that for the tier-1 supplier the list of provided and required services of the driving function represents a technical contract against which the function shall be developed.

On design level, meta-class `SoftwareClusterDesign` is used for the formalization of software that might represent such a driving function. In other words, it is assumed that a workflow exists where the design of a certain functionality on the AUTOSAR adaptive platform starts with the creation of a `SoftwareClusterDesign`.

In this case, it is further assumed that the definition of the required and provided service instances for the respective functionality is a good starting point for the development.

Please note that `SoftwareClusterDesign` supports an arbitrary complexity of software and is therefore not bound to the design of, e.g. a single driving function.

6.2 Software Cluster Design

[TPS_MANI_01112]{DRAFT} **Semantics of `SoftwareClusterDesign`** ⌈The existence of a `SoftwareClusterDesign` represents the formalized response to requirements that have initially been formulated by an OEM and that may be enriched as the development of the software progresses.

Finally, the SoftwareClusterDesign shall be taken by the integration as a further input to the definition of the result of the integration step: the definition of the SoftwareCluster.⌋*(RS_MANI_00035)*

Just to be sure, the SoftwareClusterDesign is not intended to be uploaded to the target platform. It is just an early form of the final SoftwareCluster that indeed gets uploaded. The existence of the SoftwareClusterDesign is motivated from the methodological point of view.

[constr_1557]{DRAFT} **Standardized values of SoftwareClusterDesign.category and SoftwareCluster.category** ⌈The AUTOSAR standard reserves the following values of attribute SoftwareClusterDesign.category and SoftwareCluster.category:

- ROOT_SOFTWARE_CLUSTER
- SUB_SOFTWARE_CLUSTER

⌋*()*

[TPS_MANI_01161]{DRAFT} **Impact of values of category on the semantics of SoftwareClusterDesign** ⌈A SoftwareClusterDesign of category ROOT_SOFTWARE_CLUSTER may refer to other SoftwareClusterDesigns of category SUB_SOFTWARE_CLUSTER in the role subSoftwareCluster and thereby offer a way to further break down the creation of a SoftwareClusterDesign.⌋*(RS_MANI_00035)*

[constr_1558]{DRAFT} **Existence of SoftwareClusterDesign.diagnosticAddress** ⌈The aggregation of SoftwareClusterDiagnosticAddress at SoftwareClusterDesign in the role diagnosticAddress shall only exist if the value of SoftwareClusterDesign.category is set to ROOT_SOFTWARE_CLUSTER.⌋*()*

[constr_1559]{DRAFT} **Existence of SoftwareClusterDesign.subSoftwareCluster** ⌈The Reference from SoftwareClusterDesign to itself in the role subSoftwareCluster shall only exist if the value of SoftwareClusterDesign.category is set to ROOT_SOFTWARE_CLUSTER.⌋*()*

[constr_1560]{DRAFT} **Usage of SoftwareClusterDesign.requiredARElement** ⌈The reference SoftwareClusterDesign.requiredARElement shall not be used to refer to another SoftwareClusterDesign or even SoftwareCluster.⌋*()*

Rationale for the existence of [constr_1560]: dedicated references are defined for the purpose of referring to SoftwareClusterDesigns.

[TPS_MANI_01211]{DRAFT} **Specification of executable software within SoftwareClusterDesign** ⌈One of the most prominent contents of an uploadable software package is the reference to the executable software.

Within the definition of a SoftwareClusterDesign, this reference is implicitly given by means of the reference SoftwareCluster.containedProcess.

The target of SoftwareClusterDesign.containedProcess is a ProcessDesign that represents the design-level representation of an instance (formalized as Process) of the corresponding executable program (the software image), formalized as Executable⌋*(RS_MANI_00035)*

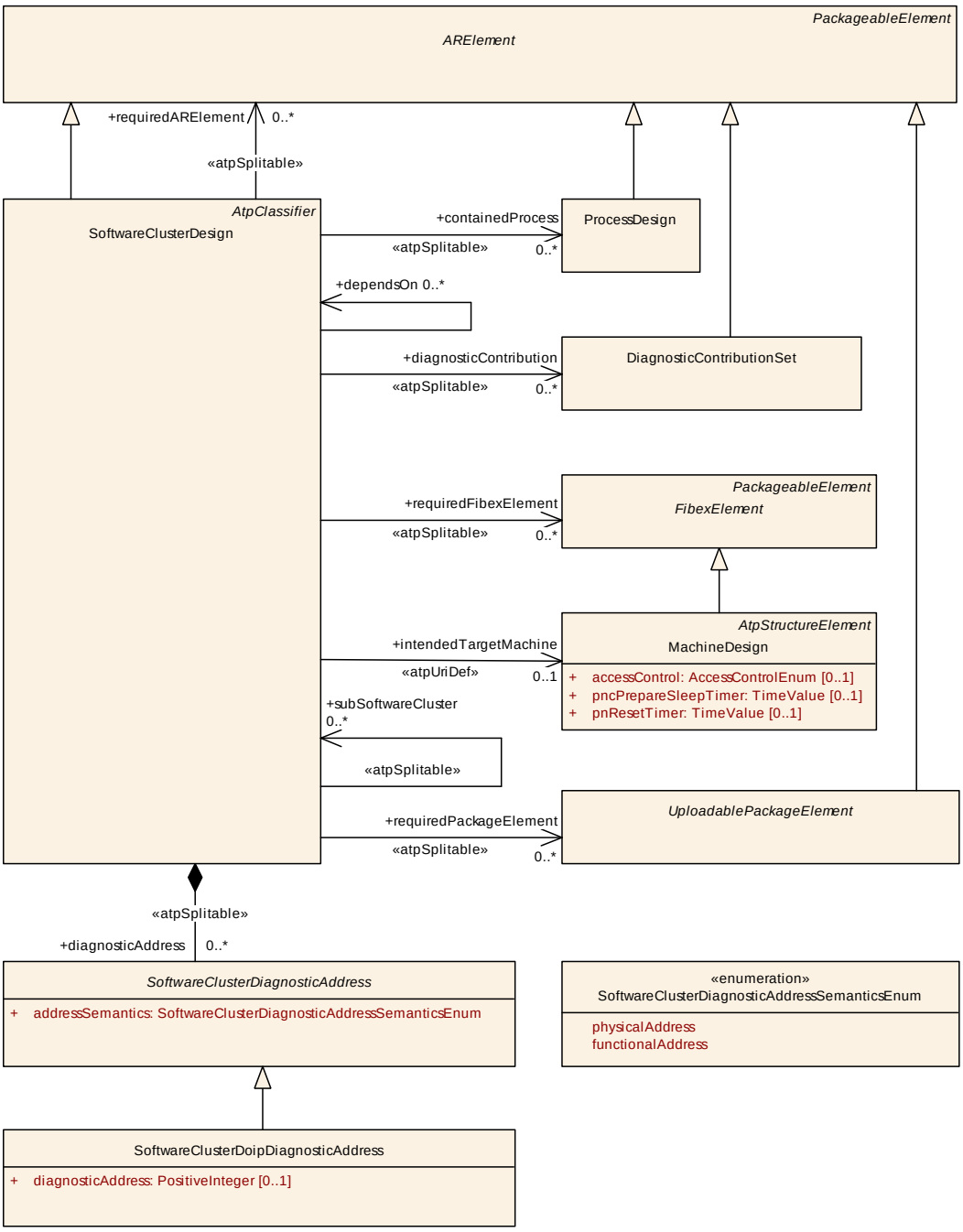


Figure 6.1: Modeling of SoftwareClusterDesign

[TPS_MANI_01113]{DRAFT} **Semantics of SoftwareClusterDesign.diagnosticAddress** ⌈The existence of the attribute SoftwareClusterDesign.diagnosticAddress can be used to express information about the distribution of diagnostic

addresses even in a very early stage of development, i.e. this is typically done by an OEM.

This includes the ability to specify multiple (i.e. several functional plus one physical) diagnostic addresses, thus the multiplicity of `diagnosticAddress` is set to 0..*.⌋*(RS_MANI_00035)*

Class	**SoftwareClusterDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign			
Note	This meta-class represents the ability for the OEM to design the grouping of software uploadable to a specific target Machine. **Tags:** atp.Status=draft atp.recommendedPackage=SoftwareClusterDesigns			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contained Process	ProcessDesign	*	ref	This reference represent the ProcessDesigns contained in the enclosing SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=containedProcess atp.Status=draft
dependsOn	SoftwareClusterDesign	*	ref	The owner SoftwareClusterDesign dependes on the referenced SoftwareClusterDesign **Tags:**atp.Status=draft
diagnostic Address	SoftwareCluster DiagnosticAddress	*	aggr	This aggregaton is used to specify the diagnsotic address. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=diagnosticAddress atp.Status=draft
diagnostic Contribution	DiagnosticContribution Set	*	ref	This reference identifes the corresponding collection of DiagnosticContributionSet. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=diagnosticContribution atp.Status=draft
intendedTarget Machine	MachineDesign	0..1	ref	This reference can be taken to identify the Machine Design for which the final SoftwareCluster shall be developed. **Stereotypes:** atpUriDef **Tags:**atp.Status=draft
required ARElement	ARElement	*	ref	This reference represents the collection of ARElements that are required for the completeness of the definition of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=requiredARElement atp.Status=draft

▽

△

Class	**SoftwareClusterDesign**			
requiredFibex Element	FibexElement	*	ref	This reference represents the collection of fibexElements that are required for the completeness of the definition of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=requiredFibexElement atp.Status=draft
required Package Element	UploadablePackage Element	*	ref	This reference points to uploadable elements that have been identified as relevant in the context of the enclosing SoftwareClusterDesign. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=requiredPackageElement atp.Status=draft
root Composition	RootSwClusterDesign ComponentPrototype	0..1	aggr	This aggregation represents the design of the software inside the SwClusterDesign terms of the communication endpoints. **Tags:**atp.Status=draft
subSoftware Cluster	SoftwareClusterDesign	*	ref	This reference is used to identify the sub-SoftwareCluster Designs of an "umbrella" SoftwareClusterDesign. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=subSoftwareCluster atp.Status=draft

Table 6.1: SoftwareClusterDesign

[TPS_MANI_01117]{DRAFT} **Semantics of `SoftwareClusterDesign.intendedTargetMachine`** ⌈The specification of `SoftwareClusterDesign.intendedTargetMachine` allows for focusing the specification of an uploadable software package to a specific `MachineDesign` from early phases of a development project.⌋*(RS_MANI_00035)*

Please note that `SoftwareCluster` doesn't have a dedicated reference to the target `Machine`.

This relation is expressed by means of a reference to `Process` that in turn can be mapped to a dedicated `Machine` by means of a `ProcessToMachineMapping`. In this context, [constr_1536] applies.

[TPS_MANI_01118]{DRAFT} **Relation between `SoftwareClusterDesign` and `DiagnosticContributionSet`** ⌈An important aspect of the definition of a `SoftwareClusterDesign` is the question what diagnostic extract shall be associated with the `SoftwareClusterDesign`.

For this purpose, a reference from `SoftwareClusterDesign` to `DiagnosticContributionSet` in the role `diagnosticContribution` is provided.

In an early stage of the development process, it is intentionally made possible to reference multiple `DiagnosticContributionSet`s in order to support the decentralized (e.g. partly done by OEM and partly done by supplier) configuration of the diagnostics stack.⌋*(RS_MANI_00035)*

[TPS_MANI_01189]{DRAFT} **Software Cluster and DiagnosticContributionSet.category** ⌈A DiagnosticContributionSet used in the context of a SoftwareCluster shall set the value of attribute category to DIAGNOSTICS_SWCL_EXTRACT.⌋*(RS_MANI_00035)*

[constr_1562]{DRAFT} **Existence of SoftwareClusterDesign.diagnosticContribution** ⌈The existence of the reference SoftwareClusterDesign.diagnosticContribution is limited to SoftwareClusterDesigns where attribute category is set to the value ROOT_SOFTWARE_CLUSTER.⌋*()*

Rationale for the existence of [constr_1562]: the definition of the diagnostic behavior is limited to the root level of a structure of SoftwareClusterDesigns in the same spirit that caused the existence of [constr_1558].

Please mind the intentionally introduced difference between SoftwareCluster and SoftwareClusterDesign in terms of the relation to DiagnosticContributionSet.

In other words, the multiplicity of the references to DiagnosticContributionSet intentionally differ.

As already explained, the SoftwareClusterDesign shall support the decentralized configuration of the DiagnosticContributionSet while the SoftwareCluster requires the existence of a final (merged) DiagnosticContributionSet.

[TPS_MANI_01119]{DRAFT} **Reference to model elements from SoftwareClusterDesign** ⌈SoftwareClusterDesign has the ability to define the following references to model elements relevant for the definition of an uploadable software package:

- references to meta-classes derived from UploadablePackageElement are formalized by way of SoftwareClusterDesign.requiredPackageElement.
- references to meta-classes derived from ARElement are formalized by way of SoftwareClusterDesign.requiredARElement.
- references to meta-classes derived from FibexElement are formalized by way of SoftwareClusterDesign.requiredFibexElement.

⌋*(RS_MANI_00035)*

Please note that the conversion of a SoftwareClusterDesign to a SoftwareCluster is not formalized by AUTOSAR. This step can be done by a tool at the discretion of the integrator.

In other words, in some cases it may be applicable to do this conversion relatively early in the development project while other projects may require to keep the SoftwareClusterDesign around for a longer period in time.

[TPS_MANI_01310]{DRAFT} **Semantics of SoftwareClusterDesign.dependsOn** ⌈The reference SoftwareClusterDesign.dependsOn can be used to prepare the definition of dependencies that exist between SoftwareClusters already on the design level.⌋*()*

In other words, the definition of SoftwareClusterDesign.dependsOn is certainly not required to build a consistent model of a SoftwareClusterDesign. The reference can only be used to “front-load” the formalization of dependencies that may later happen on the level of the design of SoftwareClusters.

6.3 Provided and required Services of Software Cluster Design

In order to support the definition of required and provided services early in the design of a SoftwareCluster[1], AUTOSAR supports the definition of a RootSwClusterDesignComponentPrototype in the context of a given SoftwareClusterDesign.

The RootSwClusterDesignComponentPrototype itself refers to a SwComponentType that in turn exposes PortPrototypes to the outside world.

Note that for the specific case of the RootSwClusterDesignComponentPrototype it is expected that the referenced SwComponentType represents a CompositionSwComponentType without any further detailing. A detailing is obviously unnecessary because the only purpose is the exposure of PortPrototypes to which AdaptivePlatformServiceInstances can be mapped.

A dedicated mapping class, ServiceInstanceToSwClusterDesignPortPrototypeMapping, is defined to support the creation of the described relation between PortPrototype and AdaptivePlatformServiceInstance.

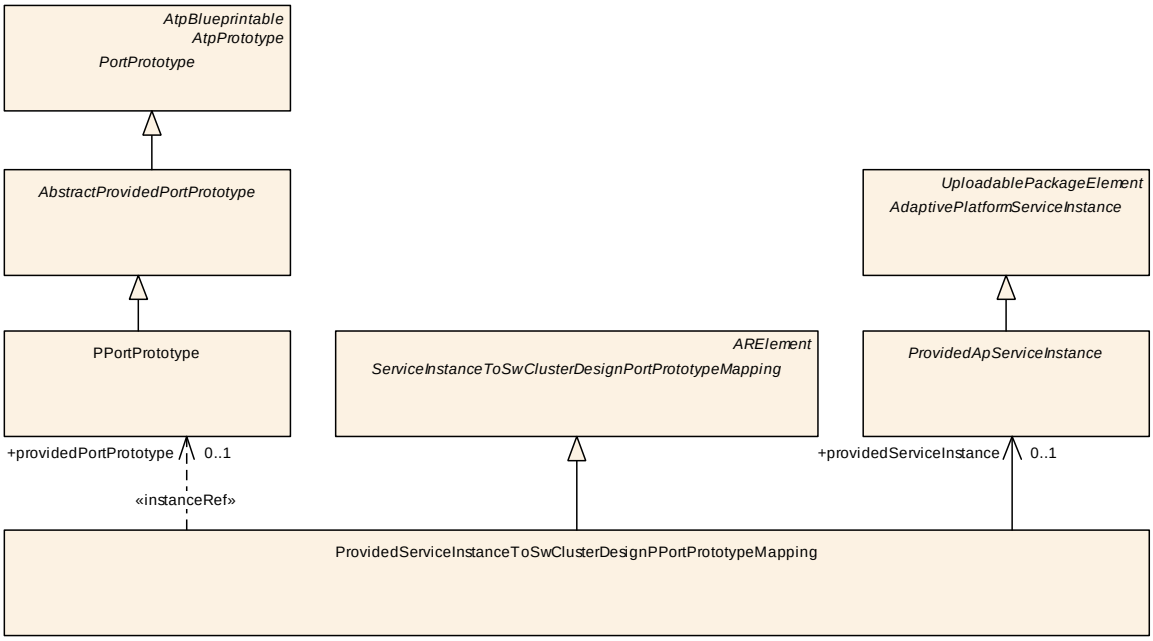


Figure 6.2: Modeling of the ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping

[1] For more information, please refer to section 14.2.

[TPS_MANI_01275]{DRAFT} **Semantics of meta-class ServiceInstanceToSwClusterDesignPortPrototypeMapping** ⌈The software-component used to type the RootSwClusterDesignComponentPrototype typically exposes a set of PortPrototypes to the outside world.

These PortPrototypes could be used for the specification of required and provided service instances. For this purpose, meta-class ServiceInstanceToSwClusterDesignPortPrototypeMapping is used.⌋*()*

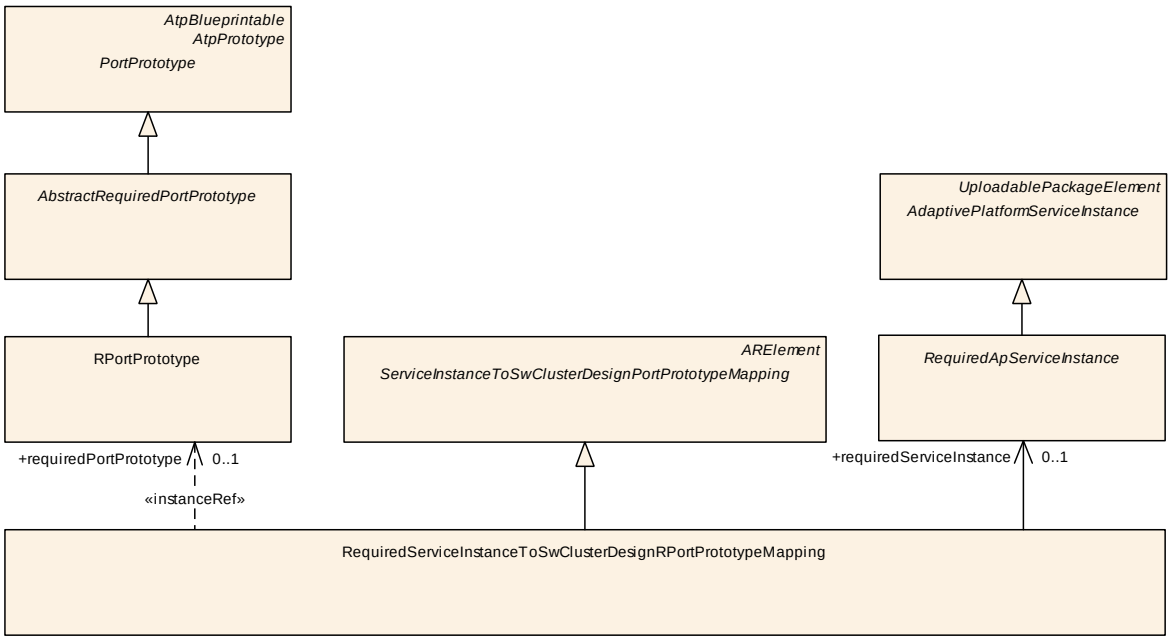


Figure 6.3: Modeling of the RequiredServiceInstanceToSwClusterDesignRPortPrototypeMapping

In Figure 6.4, the ServiceInstanceToSwClusterDesignPortPrototypeMapping is represented by a block labeled “mapping” with a circled 1. The block labeled “mapping” with a circled 2 represents the CompositionPortToExecutablePortMapping, as described in section 6.4.

Class	**RootSwClusterDesignComponentPrototype**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define the service endpoints in the scope of a SwClusterDesign. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
applicationType	SwComponentType	1	ref	This SwComponentType acts as the Type of the RootSw ClusterDesignComponentPrototype. **Tags:**atp.Status=draft

Table 6.2: RootSwClusterDesignComponentPrototype

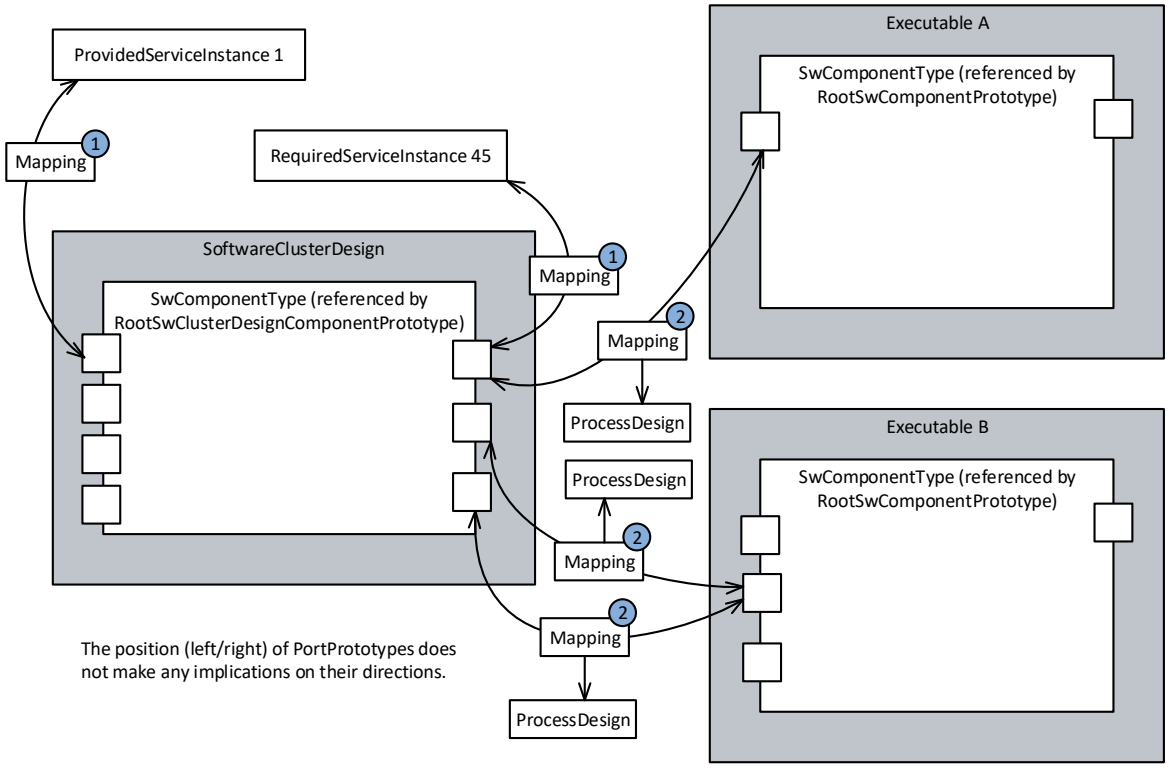


Figure 6.4: Modeling of mappings in the context of `SoftwareClusterDesign`

Class	***ServiceInstanceToSwClusterDesignPortPrototypeMapping*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow			
Note	This abstract meta-class represents the ability to assign a transport-layer-dependent ServiceInstance to a PortPrototype in the context of the SoftwareClusterDesign. With this mapping it is possible to define the list of provided and required AdaptivePlatformServiceInstances in the scope of the SoftwareCluster Design. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping, RequiredServiceInstanceToSw ClusterDesignRPortPrototypeMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 6.3: ServiceInstanceToSwClusterDesignPortPrototypeMapping

Class	**RequiredServiceInstanceToSwClusterDesignRPortPrototypeMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow

▽

△

Class	**RequiredServiceInstanceToSwClusterDesignRPortPrototypeMapping**			
Note	This concrete meta-class represents the ability to assign a transport-layer-dependent RequiredService Instance to an RPortPrototype in the context of the SoftwareClusterDesign. With this mapping it is possible to define the list of provided and required AdaptivePlatformServiceInstances in the scope of the SoftwareClusterDesign. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInstanceToSwClusterDesignPortPrototypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInstanceToSwClusterDesignPortPrototypeMapping*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
requiredPort Prototype	RPortPrototype	0..1	iref	This reference identifies the applicable PortPrototype in the scope of the SwClusterDesign. **Tags:**atp.Status=draft **InstanceRef implemented by:**RPortPrototypeInSoftware ClusterDesignInstanceRef
requiredService Instance	RequiredApService Instance	0..1	ref	Reference to a RequiredServiceInstance mapped to a given RPortPrototype in the scope of the SwCluster Design. **Tags:**atp.Status=draft

Table 6.4: RequiredServiceInstanceToSwClusterDesignRPortPrototypeMapping

Class	**ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow			
Note	This concrete meta-class represents the ability to assign a transport-layer-dependent ProvidedService Instance to a PPortPrototype in the context of the SoftwareClusterDesign. With this mapping it is possible to define the list of provided and required AdaptivePlatformServiceInstances in the scope of the Software ClusterDesign. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInstanceToSwClusterDesignPortPrototypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInstanceToSwClusterDesignPortPrototypeMapping*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
providedPort Prototype	PPortPrototype	0..1	iref	This reference identifies the applicable PortPrototype in the scope of the SwClusterDesign. **Tags:**atp.Status=draft **InstanceRef implemented by:**PPortPrototypeInSoftware ClusterDesignInstanceRef
providedService Instance	ProvidedApService Instance	0..1	ref	Reference to a ProvidedServiceInstance mapped to a given PPortPrototype in the scope of the SwCluster Design. **Tags:**atp.Status=draft

Table 6.5: ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping

6.4 Mapping of Services to Executables

A typical next step in the design workflow could be to decide about the modeling of `Executable`s inside the `SoftwareClusterDesign`. The `PortPrototype`s used

in the modeling of an Executable actually implement the endpoints to which required and provided service instances shall be mapped.

[TPS_MANI_01276]{DRAFT} **Semantics of CompositionRPortToExecutableRPortMapping and CompositionPPortToExecutablePPortMapping** ⌈In the context of the creation of an SoftwareClusterDesign, it is not possible to already define the actual mapping of PortPrototypes to AdaptivePlatformServiceInstances.

To counter this issue, and as an additional guidance for the later creation of the actual ServiceInstanceToPortPrototypeMappings it is possible to create another mapping inside the scope of the SoftwareClusterDesign that maps the PortPrototypes defined in the context of the RootSwClusterDesignComponentPrototype to the refined PortPrototypes defined in the context of Executables.⌋ *()*

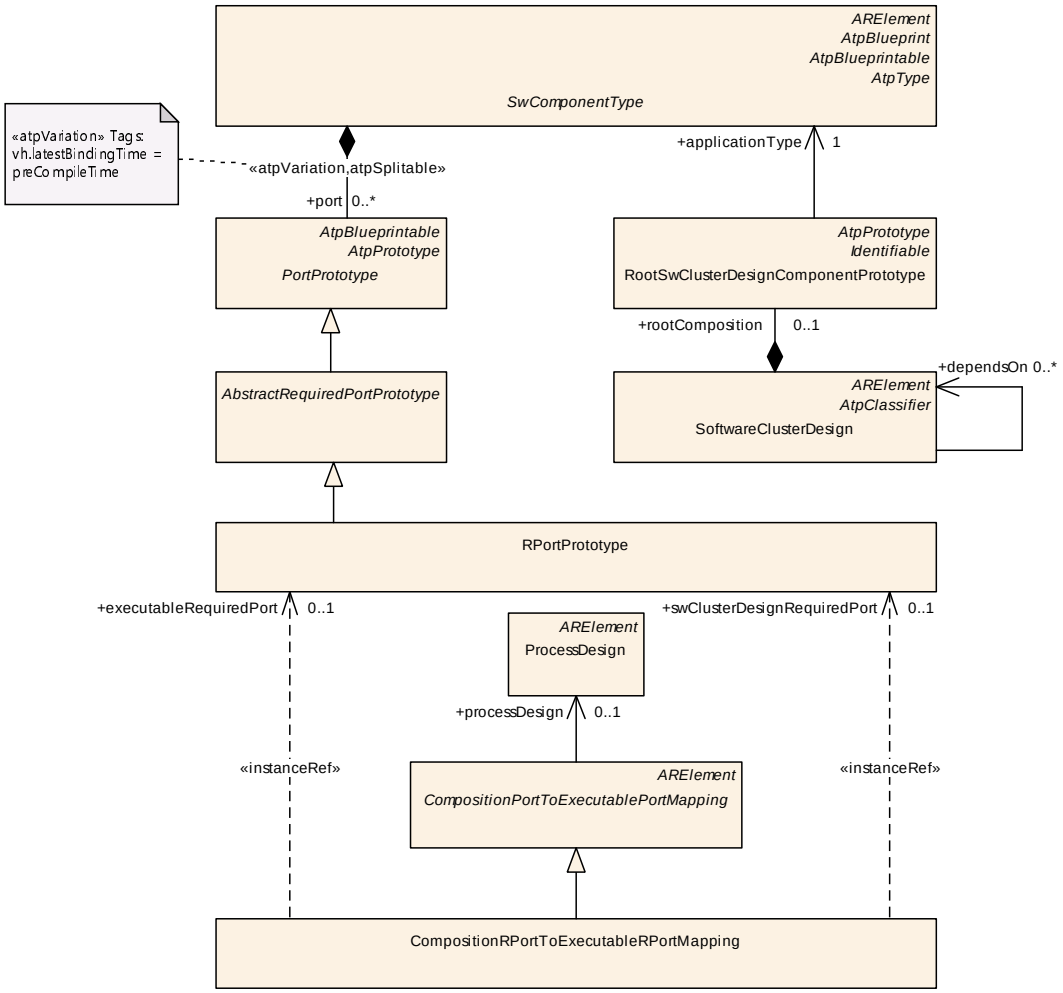


Figure 6.5: Modeling of the RootSwClusterDesignComponentPrototype and the CompositionRPortToExecutableRPortMapping

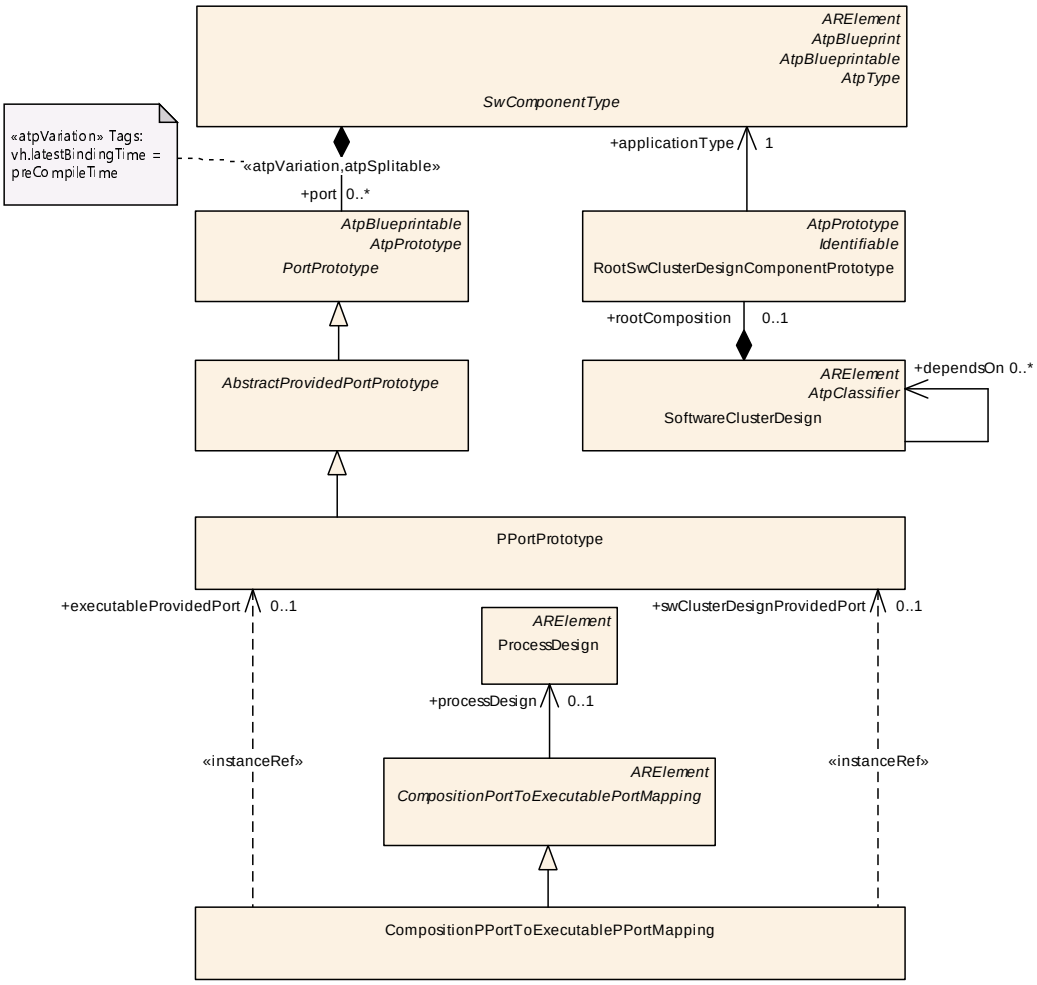


Figure 6.6: Modeling of the `RootSwClusterDesignComponentPrototype` and the `CompositionPPortToExecutablePPortMapping`

This way, it is possible to retrace the design decisions on the level of the `RootSwClusterDesignComponentPrototype` one level deeper and provide a guidance for the creation of the `ServiceInstanceToPortPrototypeMapping`, as described in section 10.2.

[TPS_MANI_01282]{DRAFT} **Semantics of reference `CompositionPortToExecutablePortMapping.processDesign`** ⌈The reference `CompositionPortToExecutablePortMapping.processDesign` identifies the applicable `ProcessDesign` for the mapping. This reference therefore disambiguates the existence of multiple `CompositionPortToExecutablePortMapping` that refer to the exact same `PortPrototype` in the context of an `Executable`.⌋*()*

The statement made by [TPS_MANI_01282] is further explained in Figure 6.4. Two `CompositionPortToExecutablePortMapping` refer to the same `PortPrototype` on the surface of the `Executable` B.

It is important to understand that each of these `CompositionPortToExecutablePortMapping`s refer to a different `ProcessDesign`.

This means that, at run-time, the two CompositionPortToExecutablePortMappings apply to different instances of the Executable B launched as different Processes (that each, in turn, refer to one of the ProcessDesigns referenced by the Executable B).

Class	***CompositionPortToExecutablePortMapping*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow			
Note	This abstract meta-class acts as a base class for the specification of a mapping between a PortPrototype owned by a RootSwClusterDesignComponentPrototype to a PortPrototype owned by a Component Prototype inside an Executable.rootSwComponentType. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	CompositionPPortToExecutablePPortMapping, CompositionRPortToExecutableRPortMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
processDesign	ProcessDesign	0..1	ref	This reference identifies the impacted ProcessDesign for this mapping. This allows for mapping multiple services to the same PortPrototype on an Executable by also referencing different ProcessDesigns. **Tags:**atp.Status=draft

Table 6.6: CompositionPortToExecutablePortMapping

Class	**CompositionRPortToExecutableRPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow			
Note	This meta-class has the ability to associate an RPortPrototype defined in the context of a SwCluster Design to an RPortPrototype in the context of an Executable. **Tags:** atp.Status=draft atp.recommendedPackage=CompositionPortToExecutablePortMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *CompositionPortToExecutablePortMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
executable RequiredPort	RPortPrototype	0..1	iref	This reference identifies the applicable PortPrototype in the context on an Executable. **Tags:**atp.Status=draft **InstanceRef implemented by:**RPortPrototypeIn ExecutableInstanceRef
swCluster DesignRequired Port	RPortPrototype	0..1	iref	This reference identifies the applicable RPortPrototype in the context of the SwClusterDesign. **Tags:**atp.Status=draft **InstanceRef implemented by:**RPortPrototypeInSoftware ClusterDesignInstanceRef

Table 6.7: CompositionRPortToExecutableRPortMapping

Class	**CompositionPPortToExecutablePPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow			
Note	This meta-class has the ability to associate a PPortPrototype defined in the context of a SwClusterDesign to a PPortPrototype in the context of an Executable. **Tags:** atp.Status=draft atp.recommendedPackage=CompositionPortToExecutablePortMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *CompositionPortToExecutablePortMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
executable ProvidedPort	PPortPrototype	0..1	iref	This reference identifies the applicable PortPrototype in the context on an Executable. **Tags:**atp.Status=draft **InstanceRef implemented by:**PPortPrototypeIn ExecutableInstanceRef
swCluster DesignProvided Port	PPortPrototype	0..1	iref	This reference identifies the applicable PPortPrototype in the context of the SwClusterDesign. **Tags:**atp.Status=draft **InstanceRef implemented by:**PPortPrototypeInSoftware ClusterDesignInstanceRef

Table 6.8: CompositionPPortToExecutablePPortMapping

7 Machine Manifest

The Machine meta-class defines the entity on which one *Adaptive AUTOSAR Software Stack* is running with an operating system. The Machine may be physical or virtual.

Some aspects of the actual Machine are already available from the System Design (see chapter 5.2) at the MachineDesign. The information defined at the MachineDesign is available to the Machine as well since Machine has a reference to the MachineDesign in the role machineDesign (see figure 5.1).

The Machine is able to aggregate one or several Processors. And each Processor consists of one or several ProcessorCores.

Meta-class ProcessorCore provides attribute coreId that can be used e.g. in a bitmask to better control the utilization of processing resources.

[constr_1549]{DRAFT} **Value of ProcessorCore.coreId** ⌈The value of ProcessorCore.coreId shall be unique in the context of the enclosing Processor.⌋*()*

An overview of the Machine meta-class is sketched in Figure 7.1.

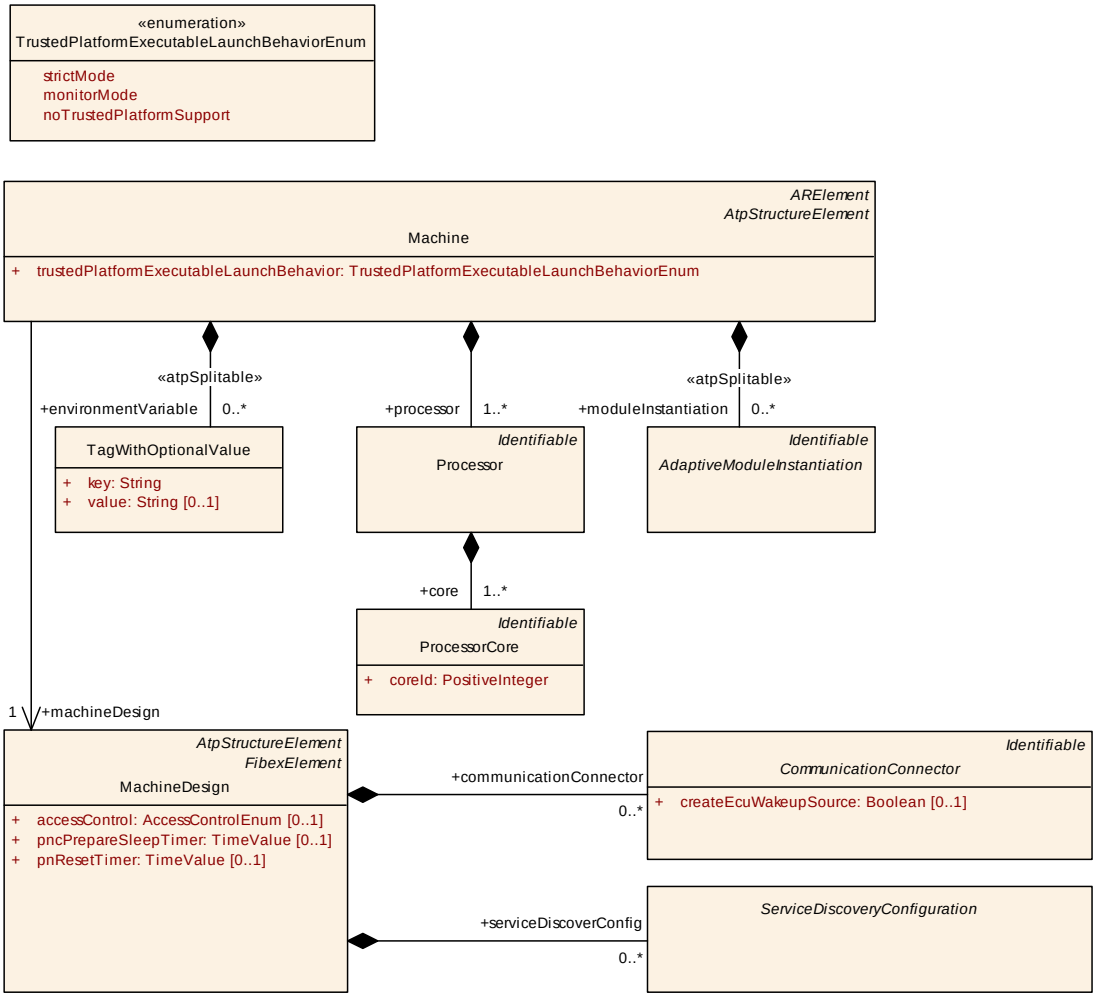


Figure 7.1: Overview about the content of the Machine configuration

[TPS_MANI_03035]{DRAFT} **Content of the Machine configuration** ⌈The purpose of the Machine is to provide machine specific configuration settings.⌋*(RS_MANI_00020, RS_MANI_00021, RS_MANI_00022, RS_MANI_00023)*

[TPS_MANI_01208]{DRAFT} **Definition of envionment variables in the scope of a Machine** ⌈It is possible to define environment variables in the scope of the entire Machine.

For this purpose the aggregation of TagWithOptionalValue in the role Machine.environmentVariable exists.

The name of the environment variable shall be specified by means of the attribute TagWithOptionalValue.key, the value can be modeled by means of TagWithOptionalValue.value.

This encloses the ability to define environment variables with empty values. For this purpose, the attribute TagWithOptionalValue.value shall simply be omitted.⌋*(RS_MANI_00022, RS_MANI_00023)*

Please note that the aggregation Machine.environmentVariable has been defined with the stereotype ≪atpSplitable≫. The consequence of this modeling is that it is possible to contribute to the definition of environment variables from **different sources**.

[TPS_MANI_01273]{DRAFT} **Support for trusted Platform** ⌈If attribute Machine.trustedPlatformExecutableLaunchBehavior is set to a value that is different from noTrustedPlatformSupport then features of the "trusted platform" are activated, depending on the concrete value of Machine.trustedPlatformExecutableLaunchBehavior.⌋*()*

Class	**Machine**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	Machine that represents an Adaptive Autosar Software Stack. **Tags:** atp.Status=draft atp.recommendedPackage=Machines			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
default Application Timeout	EnterExitTimeout	0..1	aggr	This aggration defines a default timeout in the context of a given Machine with respect to the launching and termination of applications. **Tags:**atp.Status=draft

▽

△

Class	**Machine**			
environment Variable	TagWithOptionalValue	*	aggr	This aggregation represents the collection of environment variables that shall be added to the environment defined on the level of the enclosing Machine. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=environmentVariable, environment Variable.variationPoint.shortLabel atp.Status=draft
machineDesign	MachineDesign	1	ref	Reference to the MachineDesign this Machine is implementing. **Tags:**atp.Status=draft
module Instantiation	AdaptiveModule Instantiation	*	aggr	Configuration of Adaptive Autosar module instances that are running on the machine. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=moduleInstantiation.shortName atp.Status=draft
processor	Processor	1..*	aggr	This represents the collection of processors owned by the enclosing machine. **Tags:**atp.Status=draft
secure Communication Deployment	SecureCommunication Deployment	*	aggr	Deployment of secure communication protocol configuration settings to crypto module entities. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=secureCommunicationDeployment.short Name atp.Status=draft
trustedPlatform Executable LaunchBehavior	TrustedPlatform ExecutableLaunch BehaviorEnum	1	attr	This attribute controls the behavior of how authentication affects the ability to launch for each Executable.

Table 7.1: Machine

Class	**Processor**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	This represents a processor for the execution of an AUTOSAR adaptive platform **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
core	ProcessorCore	1..*	aggr	This represents the collection of cores owned by the enclosing processor. **Tags:**atp.Status=draft

Table 7.2: Processor

Class	**ProcessorCore**
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest
Note	This meta-class represents the ability to model a processor core for the execution of an AUTOSAR adaptive platform. **Tags:**atp.Status=draft

▽

△

Class	**ProcessorCore**			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
coreId	PositiveInteger	1	attr	This attribute represents a numerical value assigned to the specific core. The value can be taken e.g. for use in a bitmask.

Table 7.3: ProcessorCore

Enumeration	**TrustedPlatformExecutableLaunchBehaviorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest
Note	This enumeration provides options for controlling the behavior of how authentication affects the ability to launch an Executable. **Tags:**atp.Status=draft
Literal	***Description***
monitorMode	An Executable shall always launch, even if the corresponding authentication fails **Tags:**atp.EnumerationLiteralIndex=1
noTrustedPlatform Support	This value shall be used if there is no TrustedPlatform support on the Machine **Tags:**atp.EnumerationLiteralIndex=2
strictMode	An Executable shall not launch if the corresponding authentication fails. **Tags:**atp.EnumerationLiteralIndex=0

Table 7.4: TrustedPlatformExecutableLaunchBehaviorEnum

7.1 Process To Machine Mapping

7.1.1 General Modeling Approach

[TPS_MANI_03147]{DRAFT} **Mapping of a `Process` to a `Machine`** ⌈The meta-class `ProcessToMachineMapping` provides the ability to map a `Process` to a `Machine`.⌋ *()*

[constr_1553]{DRAFT} **Restriction for `ProcessToMachineMapping`** ⌈The following restrictions apply for the usage of `ProcessToMachineMapping`:

1. Each combination of `Process` and `Machine` shall only be referenced by one `ProcessToMachineMapping` in the role `process` or `machine`.
2. Each `Process` shall only be referenced by a single `ProcessToMachineMapping` in the role `process`.

⌋*()*

Please note that [constr_1553] does not imply that a given `Machine` shall only be referenced by a single `ProcessToMachineMapping`. It only says that one `Process` shall only be mapped once, to exactly one `Machine`.

[constr_5004]{DRAFT} **Mapping of a `Process` to a `Machine` is mandatory in the Execution Manifest** ⌈Each `Process` shall be mapped by a `ProcessToMachineMapping` to one `Machine`.⌋*()*

[constr_5004] means that a formal description of the assignment of a `Process` to a `Machine` shall be provided in the Execution Manifest, even though the Manifest will be uploaded to the `Machine` in combination with other artifacts to which the Manifest applies. The formal `ProcessToMachineMapping` was introduced because it is useful in the processing of the model in many cases.

Please note that according to the Autosar Methodology the Execution Manifest is created on the basis of an existing Machine Manifest and therefore the link to the `Machine` can always be created in the Execution Manifest.

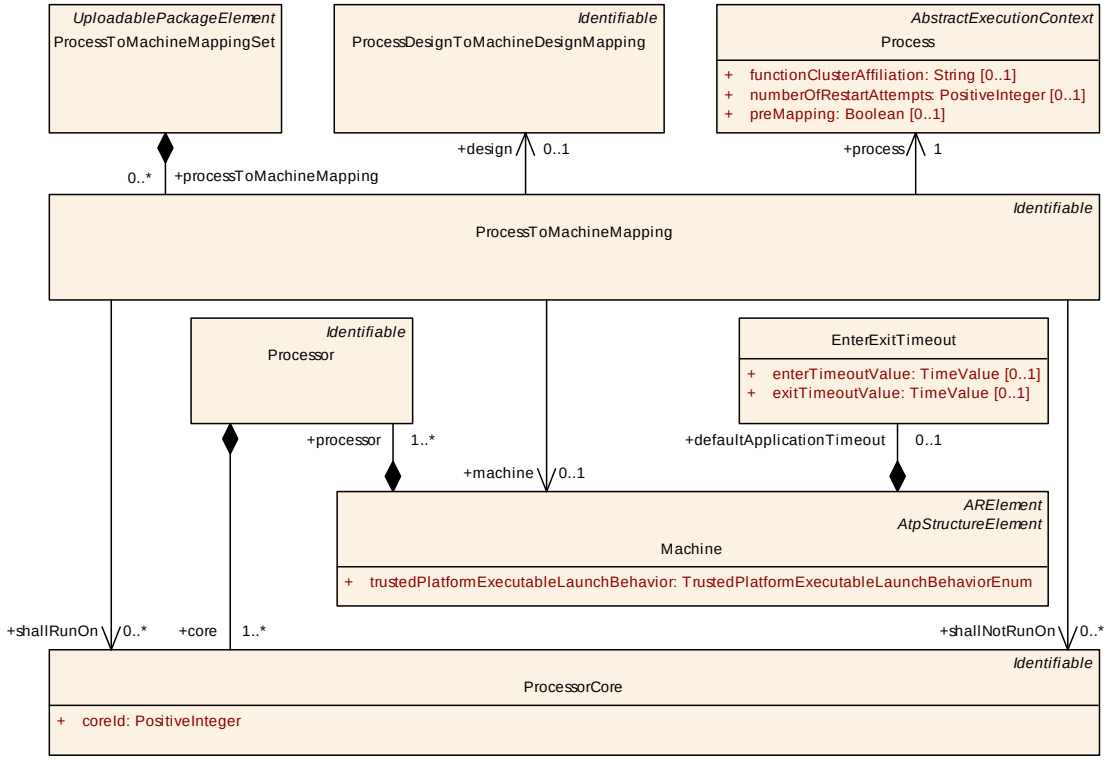


Figure 7.2: Mapping of a `Process` to a `Machine`

Class	**ProcessToMachineMappingSet**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	This meta-class acts as a bucket for collecting ProcessToMachineMappings. **Tags:** atp.Status=draft atp.recommendedPackage=ProcessToMachineMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ProcessToMachineMappingSet**			
processTo Machine Mapping	ProcessToMachine Mapping	*	aggr	This represents the collection of ProcessToMachine Mappings of the enclosing ProcessToMachineMapping Set. **Tags:**atp.Status=draft

Table 7.5: ProcessToMachineMappingSet

Class	**ProcessToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	This meta-class has the ability to associate a Process with a Machine. This relation involves the definition of further properties, e.g. timeouts. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ProcessDesignTo MachineDesignMapping	0..1	ref	This reference represents the identification of the design-time representation for the ProcessToMachine Mapping that owns the reference. **Tags:**atp.Status=draft
machine	Machine	0..1	ref	This reference identifies the Machine in the context of the ProcessToMachineMapping. **Tags:**atp.Status=draft
nonOsModule Instantiation	NonOsModule Instantiation	0..1	ref	This supports the optional case that the process represents a platform module. **Tags:**atp.Status=draft
process	Process	1	ref	This reference identifies the Process in the context of the ProcessToMachineMapping. **Tags:**atp.Status=draft
shallNotRunOn	ProcessorCore	*	ref	This reference indicates a collection of cores onto which the mapped process shall not be executing. **Tags:**atp.Status=draft
shallRunOn	ProcessorCore	*	ref	This reference indicates a collection of cores onto which the mapped process shall be executing. **Tags:**atp.Status=draft

Table 7.6: ProcessToMachineMapping

7.1.2 Core Affinity

[TPS_MANI_03148]{DRAFT} **Description of Core affinity** ⌈The meta-class `ProcessToMachineMapping` provides the ability to restrict the assignment of processes to selected `ProcessorCore`s with the two references `shallRunOn` and `shallNotRunOn`.⌋*()*

[constr_3393]{DRAFT} **Usage of `shallRunOn` and `shallNotRunOn` references** ⌈The `ProcessorCore` that is referenced by a `ProcessToMachineMapping` in the role `shallRunOn` or `shallNotRunOn` shall be aggregated by the `Machine` that is referenced in the role `machine` by the same `ProcessToMachineMapping`.⌋*()*

[constr_1676]{DRAFT} **Consistency of references `shallRunOn` and `shallNotRunOn`** ⌈Within the context of one `ProcessToMachineMapping`, all `ProcessorCore`s referenced in the role `shallRunOn` or `shallNotRunOn` shall be aggregated by the same `Processor`.⌋*()*

If a model defines that a given `Process` shall run on a select set of `ProcessorCore`s then there is hardly a use case to (in addition) also specify the opposite, i.e. that the `Process` shall not run on another set of `ProcessorCore`s, and vice versa.

In other words, either there is a motivation to identify the `ProcessorCore`s on which a `Process` is supposed to run or there is a motivation to do the exact opposite and specify the `ProcessorCore`s where the `Process` is not supposed to run.

This conclusion provides the motivation for the existence of [constr_1677].

[constr_1677]{DRAFT} **Mutual exclusive existence of references `shallRunOn` and `shallNotRunOn`** ⌈For any given `ProcessToMachineMapping`, either the reference in the role `shallRunOn` or the reference in the role `shallNotRunOn` may exist.⌋*()*

7.1.3 Default Start-up and Termination Timeout

[TPS_MANI_03151]{DRAFT} **Default value for termination timeout** ⌈The meta-class `Machine` provides the ability to define a default value for termination timeout of applications in the context of the `Machine` with the attribute `exitTimeoutValue` that is available in the `EnterExitTimeout` meta-class that is aggregated by the `Machine` in the role `defaultApplicationTimeout`.⌋*()*

[constr_3394]{DRAFT} **Default value for start-up timeout on the `Machine` is not configurable** ⌈The attribute `enterTimeoutValue` that is available in the `EnterExitTimeout` is not allowed to be used if the `EnterExitTimeout` is aggregated by the `Machine` in the role `defaultApplicationTimeout`.⌋*()*

Class	**EnterExitTimeout**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	This meta-class represents the ability to specify a pair of timeouts, one for entering, and one for exiting. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
enterTimeout Value	TimeValue	0..1	attr	This attribute represents the value of the enter timeout in seconds.
exitTimeout Value	TimeValue	0..1	attr	This attribute represents the value of the exit timeout in seconds.

Table 7.7: EnterExitTimeout

8 Execution Manifest

8.1 Overview

The purpose of the execution manifest is to provide information that is needed for the actual deployment of an application (formally modeled as an SwComponentType) onto the AUTOSAR adaptive platform.

One aspect of the deployment information is the provision of information that could in principle be provided as part of the application software code but which would make the application software code become very much bound to specific usage scenarios.

The general idea is to keep the application software code as independent as possible from the deployment scenario in order to increase the odds that the application software can be reused in different deployment scenarios.

In particular, the usage of PortPrototypes as a means to express communication with the "outside" of the application software allows for abstracting away the details (the concrete service instance identification) of the service configuration. As far as the model is concerned, the API between the application and the middleware is represented by the PortPrototype.

The application code does not use specific service instances but takes the PortPrototype as a symbolic replacement for this information. The specifics of this modeling aspect are described in section 10.

The top-level element of the Execution Manifest definition is the Process, in reference to the fact that the unit of deployment on the *AUTOSAR adaptive platform* is a binary that, at runtime, makes a POSIX process.

[TPS_MANI_01308]{DRAFT} **Process is not designed for re-usability** ⌈Meta-class Process has **not** been created with the goal of reusing it on different Machines.

However, there is *some* potential for reusing configuration aspects in the definition of the Process.stateDependentStartupConfig.startupConfig.⌋*()*

[TPS_MANI_01011]{DRAFT} **Connection between application design and application deployment** ⌈The connection between the *application design* and the *application deployment* is implemented by means of a reference from meta-class Process to meta-class Executable in the role executable.

By modeling the reference in this direction it is possible to keep the design level independent of the deployment level and, at the same time, bind the deployment to a specific design.⌋*(RS_MANI_00006)*

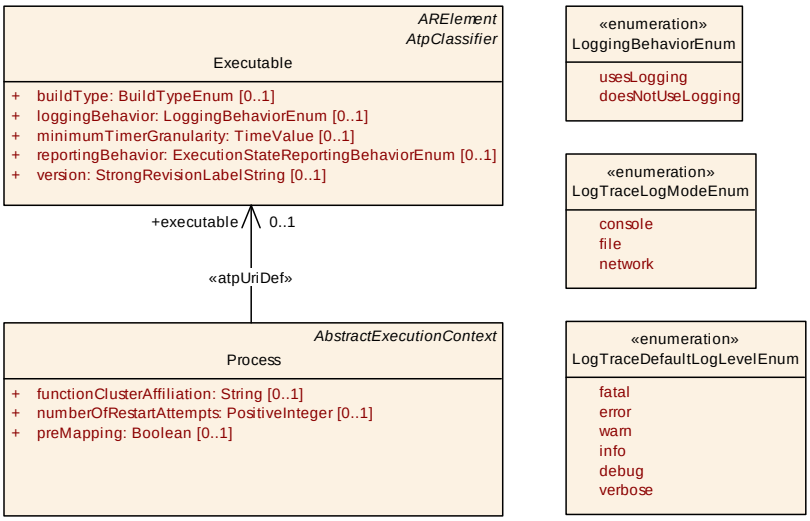


Figure 8.1: Relation of meta-classes `Executable` and `Process`

Class	**Process**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class provides information required to execute the referenced executable. **Tags:** atp.Status=draft atp.recommendedPackage=Processes			
Base	*ARElement*, *ARObject*, *AbstractExecutionContext*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ProcessDesign	0..1	ref	This reference represents the identification of the design-time representation for the Process that owns the reference. **Tags:**atp.Status=draft
deterministic Client	DeterministicClient	0..1	ref	This reference adds further execution characteristics for deterministic clients. **Tags:**atp.Status=draft
executable	Executable	0..1	ref	Reference to executable that is executed in the process. **Stereotypes:** atpUriDef **Tags:**atp.Status=draft
functionCluster Affiliation	String	0..1	attr	This attribute specifies which functional cluster the process is affiliated with.
numberOf RestartAttempts	PositiveInteger	0..1	attr	This attribute defines how often a process shall be restarted if the start fails. numberOfRestartAttempts = "0" OR Attribute not existing, start once numberOfRestartAttempts = "1", start a second time
preMapping	Boolean	0..1	attr	This attribute describes whether the executable is preloaded into the memory.
processState Machine	ModeDeclarationGroup Prototype	0..1	aggr	Set of Process States that are defined for the process. **Tags:**atp.Status=draft

△

Class	Process			
securityEvent	SecurityEventDefinition	*	ref	The reference identifies the collection of SecurityEvents that can be reported by the enclosing SoftwareCluster. **Stereotypes:** atpSplitable; atpUriDef **Tags:** atp.Splitkey=securityEvent atp.Status=draft
stateDependentStartupConfig	StateDependentStartupConfig	*	aggr	Applicable startup configurations. **Tags:**atp.Status=draft

Table 8.1: Process

[TPS_MANI_01337]{DRAFT} **Standardized values for attribute `Process.functionClusterAffiliation`** ⌈The following values of attribute `Process.functionClusterAffiliation` are standardized by AUTOSAR:

- **`STATE_MANAGEMENT`**
- **`PLATFORM_HEALTH_MANAGEMENT`**

⌋*(RS_MANI_00006)*

Please note that it is possible to use values other than from the standardized set in attribute `Process.functionClusterAffiliation`.

However, it is important that proprietary values of this attribute are formulated in a way that a potential clash with future standardized values can be avoided.

Clash-avoidance could be implemented by using a company-specific or project-specific prefix, infix, or suffix.

The `preMapping` approach of a `Process` is described in more detail in [SWS_EM_02109] in the SWS Execution Management [18].

Class	*AbstractExecutionContext* (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class acts as a base class for entities that execute code on different levels, e.g. container, process, thread, fiber. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Subclasses	Process			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 8.2: AbstractExecutionContext

Please note that the meta-model, as depicted in Figure 8.1 supports the existence of two or more `Process`es that reference the same `Executable`.

This is an indication that the specific Executable is supposed to be executed in several instances (i.e. in the form of POSIX processes) on the same platform. Such a situation is sketched in Figure 8.2

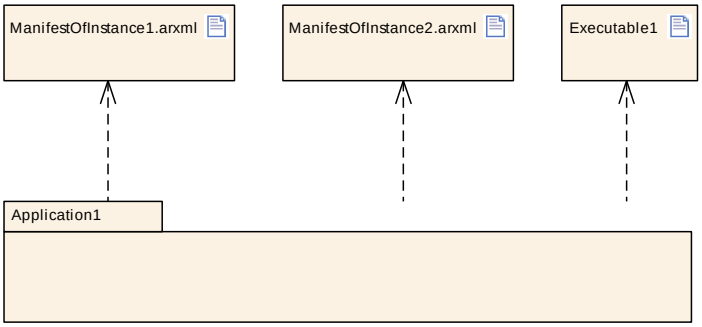


Figure 8.2: Example deployment where one Executable is bundled with two ARXML files that each contain the description of one Process

It is somehow likely that the startup conditions and startup parameters of different Processes may be different (in order to achieve a variation of the functionality of the Executable).

Therefore, it is necessary to allow for the definition of startup configurations on a per-Process-basis.

This aspect is described in section 8.2.

The supported process states that are defined in the Process.processStateMachine are described in more detail in [18].

8.2 Startup Configuration

The configuration of startup behavior is an essential part of the execution manifest.

[TPS_MANI_01012]{DRAFT} **Formal modeling of application startup behavior** ⌈The formal modeling of application startup behavior is implemented by means of the aggregation of meta-class StateDependentStartupConfig in the role Process.stateDependentStartupConfig.⌋*(RS_MANI_00007)*

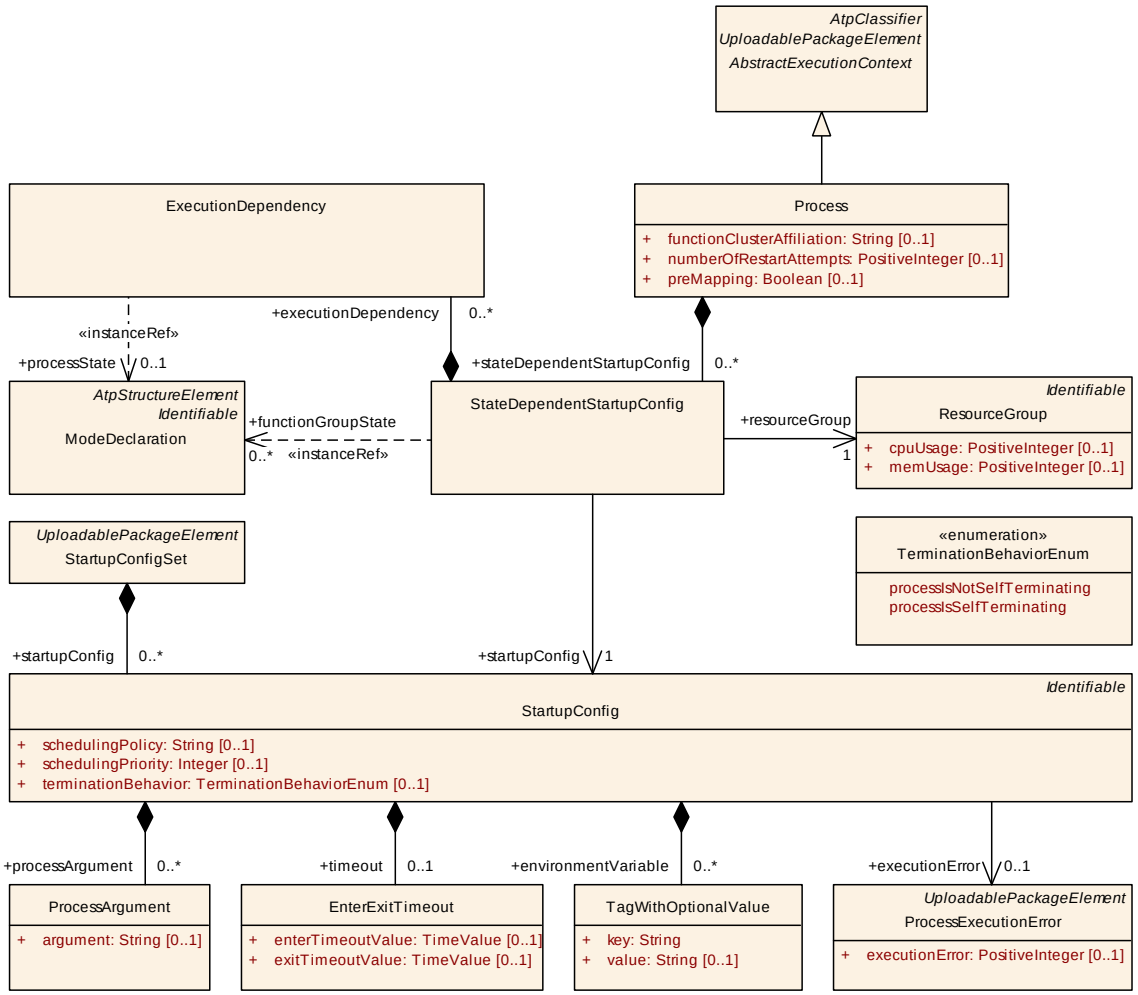


Figure 8.3: Content of a `Process`

8.2.1 State-dependent Startup Configuration

[TPS_MANI_01013]{DRAFT} **Semantics of meta-class `StateDependentStartupConfig`** ⌈The purpose of meta-class `StateDependentStartupConfig` is to qualify the startup configuration represented by meta-class `StartupConfig` for specific `ModeDeclaration`s.

In other words, the intention is to express that the `StartupConfig` is applicable if the state machines that control the startup are in the states represented by the `ModeDeclaration` referenced in the role `StateDependentStartupConfig.functionGroupState`.⌋*(RS_MANI_00007)*

As a consequence of the reference from the `StateDependentStartupConfig` to `ModeDeclaration` the `Execution Manifest` is defined for a specific `Machine` to which the binary and the Manifest is deployed.

[constr_3423]{DRAFT} **`StateDependentStartupConfig` of a `Process` shall reference a `functionGroupState`** ⌈Each `StateDependentStartupConfig` of a

Process shall reference at least one ModeDeclaration in the role functionGroupState.⌋()

However, the references to function group states within the context of one Process shall only refer to function group states **of the same function group**. This aspect is formalized by [constr_1688].

[constr_1688]{DRAFT} **StateDependentStartupConfig shall only refer to function group states of the same function group** ⌈For all StateDependentStartupConfigs aggregated in the role Process.stateDependentStartupConfig, references in the role functionGroupState to ModeDeclaration shall only refer to ModeDeclarations aggregated by the same ModeDeclarationGroup in the context of the same ModeDeclarationGroupPrototype (that represents the actual function group).⌋()

It is necessary to specify constraint [constr_3396] to regulate the number of StateDependentStartupConfigs that refer to the same ModeDeclaration in the context of one Process because the resulting startup configuration would be ambiguous.

[constr_3396]{DRAFT} **Number of Process.stateDependentStartupConfig that refer to the same functionGroupState** ⌈Within the context of a given Process, no two StateDependentStartupConfigs shall refer to the same ModeDeclaration in the role functionGroupState.⌋()

[TPS_MANI_01046]{DRAFT} **Semantics of StateDependentStartupConfig.functionGroupState** ⌈The ModeDeclarations referenced in the role StateDependentStartupConfig.functionGroupState shall be considered in a way such that the StateDependentStartupConfig applies if **any** of the referenced ModeDeclarations is active.

In other words, the ModeDeclarations are or-ed for the determination of whether a StateDependentStartupConfig is applicable.⌋(*RS_MANI_00007*)

[constr_3424]{DRAFT} **StateDependentStartupConfig shall never reference the functionGroupState Off** ⌈A StateDependentStartupConfig shall never reference the ModeDeclaration that has the shortName Off in the role functionGroupState. Please note that the Off ModeDeclaration is a special state in a Function Group as defined by [TPS_MANI_03195].⌋()

[constr_1618]{DRAFT} **Ability to shut down** ⌈In the context of one Machine, at least one Process shall have a stateDependentStartupConfig.functionGroupState that has the shortName Shutdown.⌋()

[constr_1619]{DRAFT} **Ability to restart** ⌈In the context of one Machine, at least one Process shall have a stateDependentStartupConfig.functionGroupState that has the shortName Restart.⌋()

[TPS_MANI_01209]{DRAFT} **Definition of envionment variables in process scope** ⌈It is possible to define environment variables in the scope of any given Process.

For this purpose the aggregation of TagWithOptionalValue in the role StartupConfig.environmentVariable exists.

The name of the environment variable shall be specified by means of the attribute TagWithOptionalValue.key, the value can be modeled by means of TagWithOptionalValue.value.

This encloses the ability to define environment variables with empty values. For this purpose, the attribute TagWithOptionalValue.value shall simply be omitted.⌋ *(RS_MANI_00007)*

Class	**StateDependentStartupConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class defines the startup configuration for the process depending on a collection of machine states. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
execution Dependency	ExecutionDependency	*	aggr	This attribute defines that all processes that are referenced via the ExecutionDependency shall be launched and shall reach a certain ProcessState before the referencing process is started. **Tags:**atp.Status=draft
functionGroup State	ModeDeclaration	*	iref	This represent the applicable functionGroupMode. **Tags:**atp.Status=draft **InstanceRef implemented by:**FunctionGroupStateIn FunctionGroupSetInstanceRef
resource Consumption	ResourceConsumption	0..1	aggr	This aggregation provides the ability to define resource consumption boundaries on a per-process-startup-config basis. **Tags:**atp.Status=draft
resourceGroup	ResourceGroup	1	ref	Reference to an applicable resource group. **Tags:**atp.Status=draft
startupConfig	StartupConfig	1	ref	Reference to a reusable startup configuration with startup parameters. **Tags:**atp.Status=draft

Table 8.3: StateDependentStartupConfig

[TPS_MANI_01014]{DRAFT} **Semantics of meta-class StartupConfigSet** ⌈The existence of a mode-dependent startup procedure implies the existence of a number of StartupConfigs within a given project.

Meta-class StartupConfigSet is therefore used as some sort of bucket to collect a number of StartupConfigs.⌋ *(RS_MANI_00007)*

Class	StartupConfigSet			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	Collection of reusable startup configurations for processes. **Tags:** atp.Status=draft atp.recommendedPackage=StartupConfigSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
startupConfig	StartupConfig	*	aggr	Startup configuration that is contained in the Startup ConfigSet **Tags:**atp.Status=draft

Table 8.4: StartupConfigSet

Class	StartupConfig			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class represents a reusable startup configuration for processes.. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
environment Variable	TagWithOptionalValue	*	aggr	This aggregation represents the collection of environment variables that shall be added to the respective Process's environment prior to launch. **Tags:**atp.Status=draft
executionError	ProcessExecutionError	0..1	ref	this reference is used to identify the applicable execution error **Tags:**atp.Status=draft
process Argument	ProcessArgument	*	aggr	This aggregation represents the collection of command-line arguments applicable to the enclosing StartupConfig. **Tags:**atp.Status=draft
scheduling Policy	String	0..1	attr	This attribute represents the ability to define the scheduling policy for the initial thread of the application.
scheduling Priority	Integer	0..1	attr	This is the scheduling priority requested by the application itself.
termination Behavior	TerminationBehavior Enum	0..1	attr	This attribute defines the termination behavior of the Process.
timeout	EnterExitTimeout	0..1	aggr	This aggregation can be used to specify the timeouts for launching and terminating the process depending on the StartupConfig. **Tags:**atp.Status=draft

Table 8.5: StartupConfig

[TPS_MANI_01277]{DRAFT} **Definition of a start-up timeout for a `StartupConfig` of a `Process`** ⌈Meta-class `StartupConfig` provides the ability to define a start-up timeout for a `Process` by means of the attribute `enterTimeoutValue` that is aggregated by meta-class `EnterExitTimeout` that is aggregated by the `StartupConfig` in the role `timeout`.⌋*(RS_MANI_00007)*

[TPS_MANI_01278]{DRAFT} **Definition of a termination timeout for a `StartupConfig` of a `Process`** ⌈Meta-class `StartupConfig` provides the ability to define a termination timeout for a `Process` by means of the attribute `exitTimeoutValue` that is aggregated by meta-class `EnterExitTimeout` that is aggregated by the `StartupConfig` in the role `timeout`.⌋*(RS_MANI_00007)*

8.2.2 Scheduling

[TPS_MANI_01061]{DRAFT} **Requirements on scheduling** ⌈The attributes `StartupConfig.schedulingPolicy` and `StartupConfig.schedulingPriority` make requirements on the scheduling of the main thread of a process that is created out of launching the corresponding `Executable`.⌋*(RS_MANI_00007)*

[TPS_MANI_01328]{DRAFT} **Standardized values for attribute `StartupConfig.schedulingPolicy`** ⌈The following values are standardized for attribute `StartupConfig.schedulingPolicy`:

- **`SCHED_RR`**
- **`SCHED_FIFO`**
- **`SCHED_OTHER`**

⌋*()*

It is possible to use a custom, non-standardized value for the attribute `StartupConfig.schedulingPolicy` but this option comes with the obligation to use a value that is guaranteed to not clash with possible future extensions of the collection of standardized values.

[TPS_MANI_01188]{DRAFT} **Semantics of attribute `schedulingPriority`** ⌈The value of attribute `StartupConfig.schedulingPriority` shall be interpreted such that the higher values represent a higher scheduling priority.⌋*(RS_MANI_00007)*

[constr_1692]{DRAFT} **Value of `schedulingPriority`** ⌈The value of attribute `StartupConfig.schedulingPriority` shall be set to a positive integer value.⌋*()*

8.2.3 Process Arguments

Please find more information about the interpretation of `ProcessArgument` in the SWS Execution Manifest [18].

[constr_1769]{DRAFT} **Existence of `ProcessArgument.argument`** ⌈For each `ProcessArgument`, attribute `argument` shall exist **at the time when manifest creation is finished**.⌋*()*

Class	**ProcessArgument**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class has the ability to define command line arguments for processing by the Main function. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
argument	String	0..1	attr	This represents one command-line argument to be processed by the executable software.

Table 8.6: ProcessArgument

8.2.4 Association with Resource Group

Meta-class `StateDependentStartupConfig` also supports the specification of a relation to a resource group.

[TPS_MANI_01017]{DRAFT} **Relation of startup configuration to resource group** ⌈The modeling of a resource group is possible by means of meta-class `ResourceGroup` in the `OsModuleInstantiation` of the `Machine` and the assignment of a `Process` to a `ResourceGroup` is supported by the association from `StateDependentStartupConfig` to `ResourceGroup` in the role `resourceGroup`.⌋*(RS_MANI_00007)*

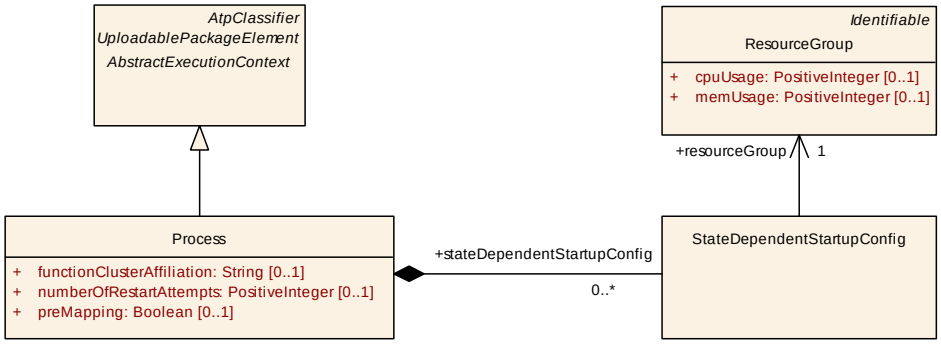


Figure 8.4: Modeling of how `Process` relates to `ResourceGroup`

[constr_3413]{DRAFT} **`StateDependentStartupConfig` of a `Process` is mapped to exactly one `ResourceGroup`** ⌈Each `StateDependentStartupConfig` of a `Process` shall be assigned to exactly one `ResourceGroup` that is defined in the Machine Manifest.⌋*()*

8.2.5 Execution Dependency

The modeling of an execution dependency makes two `Process`es become associated to each other by means of the definition of an `ExecutionDependency`.

But since the reference that defines the execution dependency is modeled as an ≪instanceRef≫ the referenced Process needs to be extracted from the context references in the ≪instanceRef≫.

Once the two Processes are identified it is necessary for the validity of the startup dependency that they refer to the identical function group.

[TPS_MANI_01041]{DRAFT} **Startup configuration supports the definition of a launch sequence dependency** ⌈The modeling of startup configuration also supports the definition of a launch sequence dependency, formalized by the meta-class ExecutionDependency that is aggregated by StateDependentStartupConfig in the role executionDependency.

The ExecutionDependency allows to define a dependency to a process that needs to be in a specific process state before the process that aggregates the ExecutionDependency via StateDependentStartupConfig is launched.⌋*(RS_MANI_00007)*

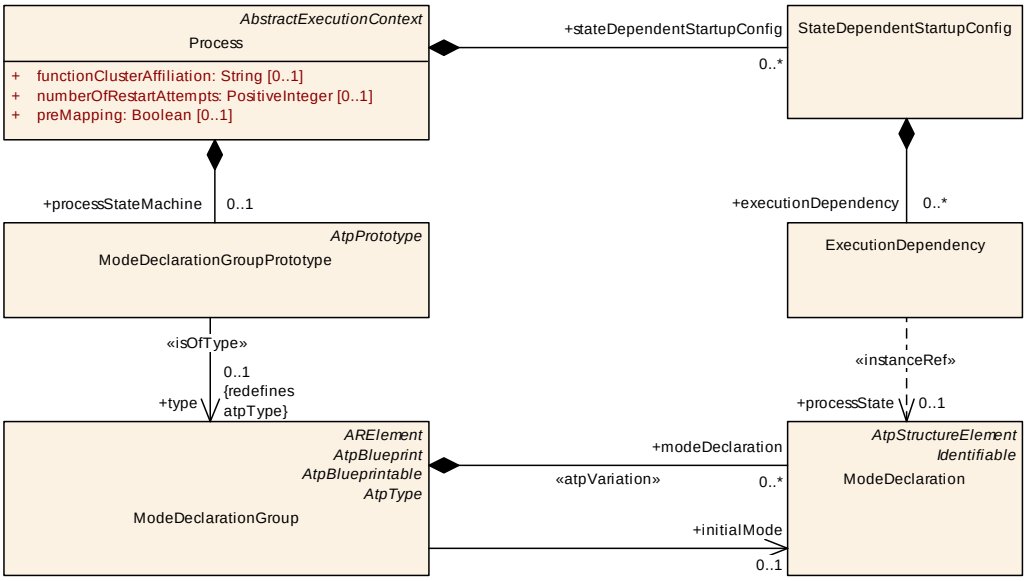


Figure 8.5: Modeling of how Process relates to ModeDeclaration owned by another Process

[constr_1689]{DRAFT} **Modeling of a startup dependency between different Processes** ⌈The existence of attribute Process.stateDependentStartupConfig.executionDependency is only valid if the owner of the stateDependentStartupConfig.executionDependency (in other words: the referencing Process) and the owner of the ModeDeclarationGroupPrototype referenced in the role contextModeDeclarationGroupPrototype within the reference stateDependentStartupConfig.executionDependency.processState (i.e. the referenced Process) refer to the identical function group state formalized as ModeDeclaration.⌋*()*

Figure 8.6 provides an exemplary explanation of [constr_1689]. In this example, Process “B” (the referencing Process as of [constr_1689]) defines an executionDependency to Process “A”.

This executionDependency is only valid if both Process “A” and Process “B” aggregate a StateDependentStartupConfig that refers to the same function group state “MD” within function group “FG”.

Process “A” can be found by following the ExecutionDependency (specifically the contextModeDeclarationGroupPrototype) and the ≪instanceRef≫ that goes from the ExecutionDependency to the Process State “PS”.

The **owner** of “PS” is Process “B”, and if “B” refers to function group state “MD” within function group “FG” and if “A” refers to function group state“MD” within “FG” then the constraint [constr_1689] is fulfilled.

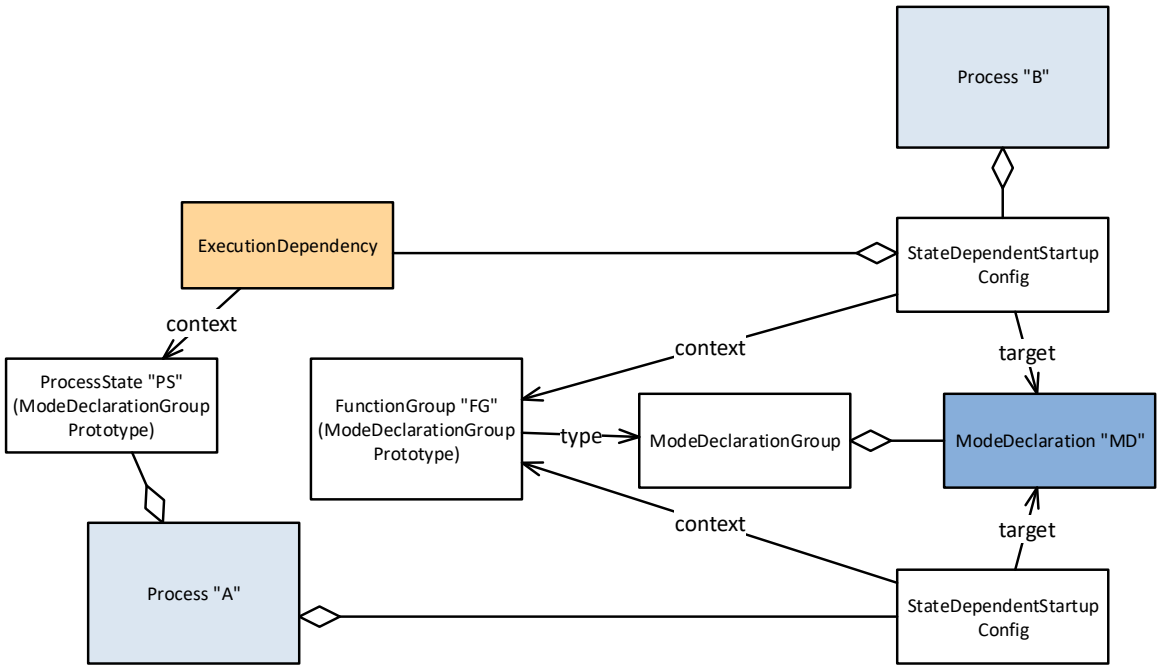


Figure 8.6: Explanation of dependencies from one Process to another

[constr_1744]{DRAFT} **Definition of process state In the context of the ExecutionDependency** ⌈The target ModeDeclaration referenced in the role ExecutionDependency.processState shall fulfill the following conditions:

- It shall be owned by a ModeDeclarationGroup that is referenced by a ModeDeclarationGroupPrototype (in the role type) that in turn shall be aggregated by a Process.
- The shortName of the ModeDeclaration has either of the following values:
  - Running
  - Terminated

⌋()

Class	**ExecutionDependency**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This element defines a ProcessState in which a dependent process needs to be before the process that aggregates the ExecutionDependency element can be started. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
processState	ModeDeclaration	0..1	iref	This represent the applicable modeDeclaration that represents an ProcessState. **Tags:**atp.Status=draft **InstanceRef implemented by:**ModeInProcessInstanceRef

Table 8.7: ExecutionDependency

[constr_1606]{DRAFT} **Processes with mutual ExecutionDependencys** ⌈A Process.stateDependentStartupConfig.executionDependency shall not refer to any ModeDeclaration owned by a second Process that in turn refers via stateDependentStartupConfig.executionDependency to any ModeDeclaration owned by the first Process.⌋()

8.2.6 Assignment of Processes to Function Group states

There are use cases where starting and terminating of individual groups of processes is necessary. This is supported in AUTOSAR by function groups that group processes together.

A function group may have a number of function group states, e.g. Running, Idle, Terminating. The StateDependentStartupConfig of a Process can be assigned to a function group state and the start-up of the Process will then depend on this assignment.

The modeling of a function group and its function group states is described in section 8.4 in more detail. The usage of Function Groups is described in more detail in [18].

[TPS_MANI_03152]{DRAFT} **Assignment of a StateDependentStartupConfig to a function group state** ⌈The StateDependentStartupConfig is assigned to a function group state with the functionGroupState reference.⌋*(RS_MANI_00041)*

8.2.7 Resource Consumption Boundaries

[TPS_MANI_01269]{DRAFT} **Specification of boundaries for resource consumption** ⌈It is possible to specify boundaries for resource consumption, specifically in terms of memory consumption for system memory and heap, of a given startup configuration of a Process:

- The formalization of heap usage is represented by meta-class HeapUsage, aggregated via meta-class ResourceConsumption at StateDependentStartupConfig. The actual value of the heap usage is computed out of the sum of all aggregated ResourceConsumption.heapUsage.
- The formalization of system (i.e. kernel-space) memory usage is represented by meta-class SystemMemoryUsage, aggregated via meta-class ResourceConsumption at StateDependentStartupConfig. The actual value of the system memory usage is computed out of the sum of all aggregated ResourceConsumption.systemMemoryUsage.

⌋()

Please note the difference between the ability of defining resource consumption boundaries for a single Process, as opposed to the ability to associate a Process with a ResourceGroup that has the ability to also define resource consumption boundaries, albeit on a more coarse-grained level.

In contrast to that, the StateDependentStartupConfig.resourceConsumption allows for a fine-grained definition that can even observe the differences in resource consumption with respect to different startup configurations.

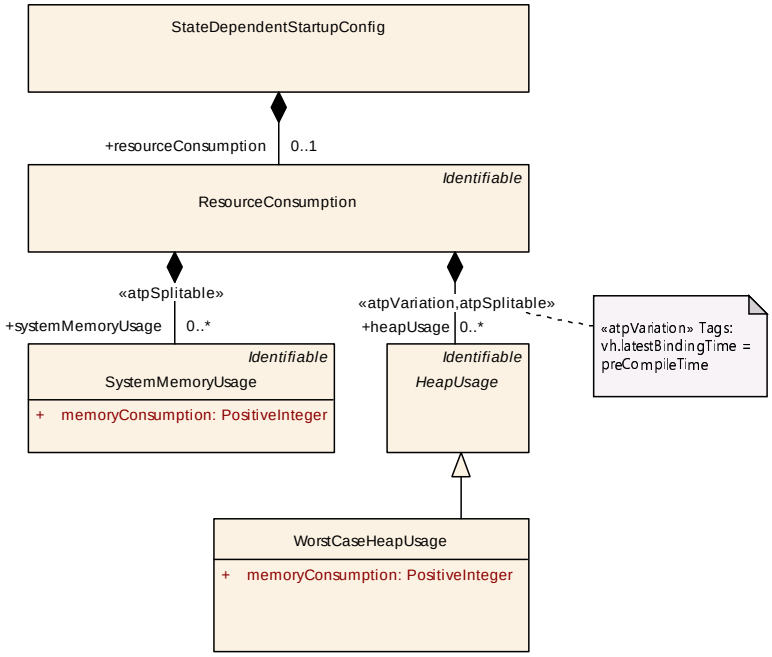


Figure 8.7: Modeling of resource consumption boundaries for a given Process

[constr_1707]{DRAFT} **Eligible subclasses of HeapUsage in the context of StateDependentStartupConfig.resourceConsumption** ⌈The definition of StateDependentStartupConfig.resourceConsumption.heapUsage shall only be done by means of the concrete sub-class WorstCaseHeapUsage.⌋()

Class	ResourceConsumption			
Package	M2::AUTOSARTemplates::CommonStructure::ResourceConsumption			
Note	Description of consumed resources by one implementation of a software.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
heapUsage	HeapUsage	*	aggr	Collection of the heap memory allocated by this implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=heapUsage.shortName, heapUsage.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
systemMemoryUsage	SystemMemoryUsage	*	aggr	Collection of the system memory allocated by the owner. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=systemMemoryUsage.shortName atp.Status=draft

Table 8.8: ResourceConsumption

Class	**HeapUsage** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::ResourceConsumption::HeapUsage			
Note	Describes the heap memory usage of a SW-Component.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	WorstCaseHeapUsage			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 8.9: HeapUsage

Class	WorstCaseHeapUsage			
Package	M2::AUTOSARTemplates::CommonStructure::ResourceConsumption::HeapUsage			
Note	Provides a formal worst case heap usage.			
Base	*ARObject*, *HeapUsage*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
memoryConsumption	PositiveInteger	1	attr	Worst case heap consumption. Unit: byte.

Table 8.10: WorstCaseHeapUsage

Class	SystemMemoryUsage			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	Describes the system memory (i.e. kernel space) consumption. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	SystemMemoryUsage			
memory Consumption	PositiveInteger	1	attr	Provides a formal worst case system usage.

Table 8.11: SystemMemoryUsage

8.2.8 Error and Termination Behavior

[TPS_MANI_01334]{DRAFT} **Semantics of `StartupConfig.terminationBehavior`** ⌈The attribute `StartupConfig.terminationBehavior` defines the termination behavior of the `Process` in terms of whether (or not) the `Process` that references the enclosing `StartupConfig` in the role `stateDependentStartupConfig`. `startupConfig` is configured to self-terminate.⌋*()*

Enumeration	TerminationBehaviorEnum
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest
Note	This enumeration provides options for controlling of how a Process terminates. **Tags:**atp.Status=draft
Literal	Description
processIsNotSelf Terminating	The Process terminates only on request from Execution Management. **Tags:**atp.EnumerationLiteralIndex=0
processIsSelf Terminating	The Process is allowed to terminate without request from Execution Management. **Tags:**atp.EnumerationLiteralIndex=1

Table 8.12: TerminationBehaviorEnum

[constr_10007]{DRAFT} **Existence of `ProcessExecutionError.executionError`** ⌈For each `ProcessExecutionError`, attribute `executionError` shall exist **at the time when manifest creation is finished**.⌋*()*

[constr_10008]{DRAFT} **Value of `ProcessExecutionError.executionError`** ⌈The value of attribute `ProcessExecutionError.executionError` shall at least be set to 1 (or higher).⌋*()*

Class	ProcessExecutionError			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class has the ability to describe the value of a execution error along with a documentation of its semantics. **Tags:** atp.Status=draft atp.recommendedPackage=ProcessExecutionErrors			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	Type	Mult.	Kind	Note

▽

△

Class	**ProcessExecutionError**			
executionError	PositiveInteger	0..1	attr	This attribute defines the numeric value which Execution Management and Platform Health Management reports to State Management if the Process terminates unexpectedly or violates its supervision. It shall give further error information for error recovery.

Table 8.13: ProcessExecutionError

8.3 Deterministic Client

As already explained in section 3.19.1, there is a use case to support the concept of the so-called `Deterministic Client` on the *AUTOSAR adaptive platform*. The conceptual background of `Deterministic Client` is explained in the SWS Execution Management [18].

The support for this concept consists of two aspects. The *design aspect* has already been explained in section 3.19.1 while the *deployment aspect* is discussed in this chapter.

[TPS_MANI_01203]{DRAFT} **Semantics of `DeterministicClient`** ⌈The existence of reference `Process.deterministicClient` means that the enclosing `Process` implements the concept of a `Deterministic Client`.

Further information for the configuration of the `Deterministic Client` can be obtained from the `ProcessDesign` referenced in the role `Process.design`.⌋*(RS_MANI_00050)*

The details of the support for `Deterministic Client` are visualized in Figure 8.8.

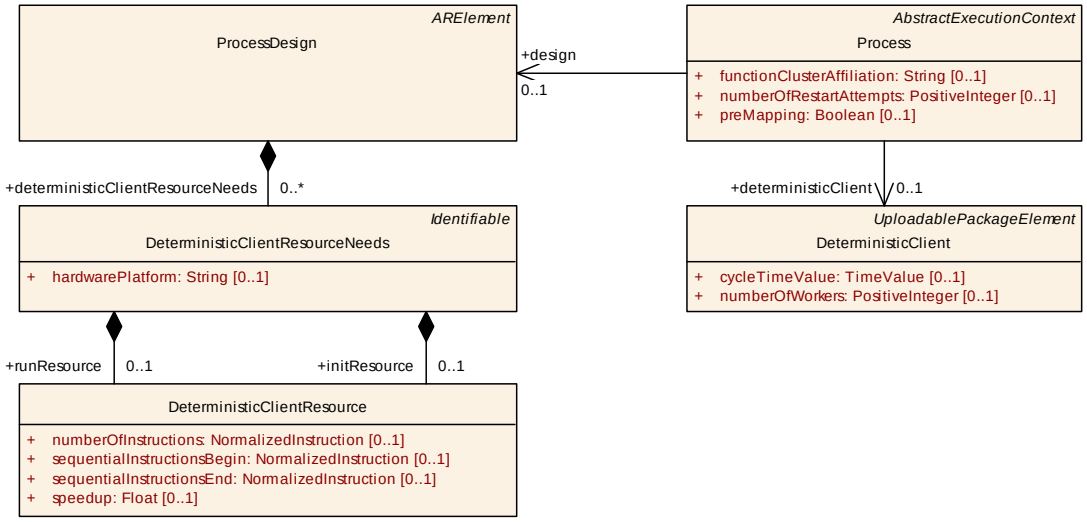


Figure 8.8: Modeling of support for `Deterministic Client` in the deployment

Class	**DeterministicClient**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	The meta-class DeterministicClient provides the ability to support the deterministic execution of one or more processes with specific configuration parameters for DeterministicClient library functions. **Tags:** atp.Status=draft atp.recommendedPackage=DeterministicClients			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cycleTimeValue	TimeValue	0..1	attr	This attribute represents the cycle time for execution of a DeterministicClient activation cycle.
numberOf Workers	PositiveInteger	0..1	attr	Number of independent workers that process data-sets. Size of the worker pool shall be decided based on availability of resources like processor cores or memory.

Table 8.14: DeterministicClient

8.4 Function Groups

8.4.1 Semantics of Function Group

Function groups with function group states individually control groups of functionally coherent Application processes. The `Process` state may depend on a mode that is defined in the function group in case that the `StateDependentStartupConfig` refers to the function group state with the `functionGroupState` reference.

The usage of Function Groups is described in more detail in [18].

[TPS_MANI_03145]{DRAFT} **Description of a function group** ⌈By defining a `ModeDeclarationGroupPrototype` aggregated in the role `FunctionGroupSet.functionGroup` it is possible to define a function group that has a `shortName` and a set of Modes (States).

The `ModeDeclarationGroupPrototype` points to a reusable `ModeDeclarationGroup` in the role `type` that contains the different modes as `ModeDeclaration`s and a designated `initialMode`.⌋*(RS_MANI_00041)*

[TPS_MANI_03194]{DRAFT} **Function Group State** ⌈A function group state is described by a `ModeDeclaration` within a `ModeDeclarationGroup` that is referenced in the role `type` by a `ModeDeclarationGroupPrototype` aggregated as `functionGroup` by a `FunctionGroupSet`. The function group state is identified by its `shortName`.⌋*(RS_MANI_00041)*

The modeling described in [TPS_MANI_03145] and [TPS_MANI_03194] is depicted in Figure 8.9.

[TPS_MANI_03195]{DRAFT} **Off state in Function Group** ⌈Each `functionGroup` shall define a `ModeDeclaration` with the `shortName` `Off`. This `ModeDeclaration`

shall also be referenced in the role initialMode by ModeDeclarationGroup that types the respective functionGroup.⌋*(RS_MANI_00041)*

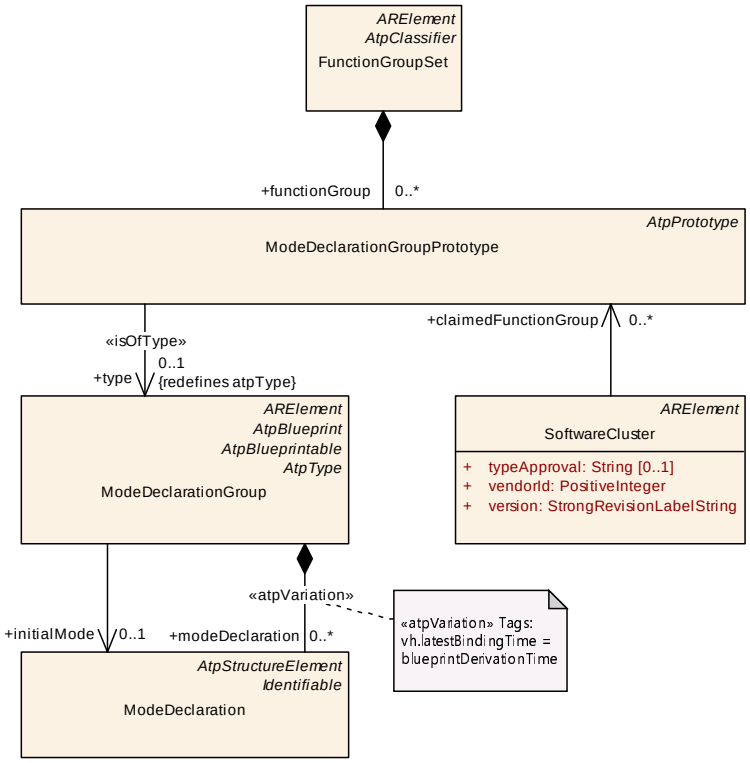


Figure 8.9: Configuration of Function Groups

[constr_1786]{DRAFT} **Restriction to use functionGroup in terms of SoftwareCluster** ⌈Each functionGroup shall only be referenced in the role claimedFunctionGroup by **at most one** SoftwareCluster.⌋*()*

[constr_1787]{DRAFT} **Restricted use of function groups in the context of a SoftwareCluster** ⌈All Processes referenced by a SoftwareCluster in the role containedProcess shall only aggregate StateDependentStartupConfigs where the reference functionGroupState refers to a ModeDeclarationGroupPrototype (as context) that is also referenced by the same SoftwareCluster in the role claimedFunctionGroup.⌋*()*

The description of SoftwareCluster can be found in section 14.

[constr_10023]{DRAFT} **Mandatory content of any functionGroup** ⌈All ModeDeclarationGroupPrototypes aggregated by a FunctionGroupSet in the role functionGroup shall refer to a ModeDeclarationGroup that contains one ModeDeclaration with the shortName **Verify**.⌋*()*

Class	FunctionGroupSet			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class provides the ability to create arbitrary collections of function groups. **Tags:** atp.Status=draft atp.recommendedPackage=FunctionGroupSets			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
functionGroup	ModeDeclarationGroup Prototype	*	aggr	This aggregation represents the collection of function groups. **Tags:**atp.Status=draft

Table 8.15: FunctionGroupSet

Class	ModeDeclarationGroupPrototype			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	The ModeDeclarationGroupPrototype specifies a set of Modes (ModeDeclarationGroup) which is provided or required in the given context.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
type	ModeDeclarationGroup	0..1	tref	The "collection of ModeDeclarations" (= ModeDeclaration Group) supported by a component **Stereotypes:** isOfType

Table 8.16: ModeDeclarationGroupPrototype

Class	ModeDeclarationGroup			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	A collection of Mode Declarations. Also, the initial mode is explicitly identified. **Tags:**atp.recommendedPackage=ModeDeclarationGroups			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
initialMode	ModeDeclaration	0..1	ref	The initial mode of the ModeDeclarationGroup. This mode is active before any mode switches occurred.
mode Declaration	ModeDeclaration	*	aggr	The ModeDeclarations collected in this ModeDeclaration Group. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=blueprintDerivationTime
modeTransition	ModeTransition	*	aggr	This represents the avaliable ModeTransitions of the ModeDeclarationGroup

Table 8.17: ModeDeclarationGroup

Class	**ModeDeclaration**			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 8.18: ModeDeclaration

8.4.2 Machine Function Group

Please note that one `functionGroup` claimed by one `SoftwareCluster` of `category` `PLATFORM_CORE` takes the role of a "machine function group".

This `functionGroup` is required to have a dedicated `shortName` and it also is required to define a certain minimal, but extensible set of `ModeDeclaration`s that also have standardized `shortName`s.

[TPS_MANI_01330]{DRAFT} **Definition of machine function group** ⌈Exactly one `functionGroup` shall exist that has the `shortName` "MachineFG" and that is typed by a `ModeDeclarationGroup` that defines at least the following list of `ModeDeclaration`s with the `shortName`s

- **`Off`**,
- **`Verify`**,
- **`Startup`**,
- **`Shutdown`**, and
- **`Restart`**.

⌋*(RS_MANI_00041)*

Please note that the startup of a `Process` may depend on Modes that are defined in the context of a `SoftwareCluster` of `category` `PLATFORM_CORE`. The `StateDependentStartupConfig` is described in chapter 8.2.

[constr_1789]{DRAFT} **Scope of machine function group** ⌈The `functionGroup` that represents the machine function group (see [TPS_MANI_01330]) shall only be referenced in the role `claimedFunctionGroup` by a `SoftwareCluster` of `category` `PLATFORM_CORE`.⌋*()*

8.5 Reporting of Security Events

It is possible to report so-called security events (formalized by meta-class `SecurityEventDefinition`) from the context of a `Process`.

This approach works for application-level software as well as for functional clusters with the exception of the Execution Manager (because the Execution Manager is itself not modeled as a `Process`).

Please find more information about the semantics and usage of security events in the TPS Security Extract Template [21].

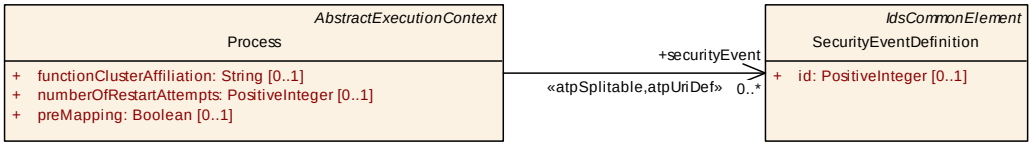


Figure 8.10: Modeling of support for the reporting of `SecurityEventDefinition` on deployment level

Class	**SecurityEventDefinition**			
Package	M2::AUTOSARTemplates::SecurityExtractTemplate			
Note	This meta-class defines a security-related event as part of the intrusion detection system. **Tags:** atp.Status=draft atp.recommendedPackage=SecurityEventDefinitions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *IdsCommonElement*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventSymbol Name	SymbolProps	0..1	aggr	This aggregation defines optionally an alternative Event Name for the SecurityEventDefinition in case there is a collision of shortNames. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=eventSymbolName.shortName atp.Status=draft
id	PositiveInteger	0..1	attr	This attribute represents the numerical identification of the defined security event. The identification shall be unique within the scope of the IDS. **Tags:**atp.Status=draft

Table 8.19: SecurityEventDefinition

9 Platform Module Development

The model of platform modules and their instantiation has two major use-cases:

- provide dedicated attributes to configure the platform modules
- define the potential start of the module's executable as process.

The two use-cases are combined in one modeling approach: the Machine.moduleInstantiation, which collects sub-classes of AdaptiveModuleInstantiation. This modeling approach boils down to the variety of platform module models found in this chapter.

The OsModuleInstantiation defines several attributes to be configured for the Os, however the OsModuleInstantiation is the only AdaptiveModuleInstantiation where it is not possible to map it to a Process model element.

Of course there will be processes running the Os on the Machine anyway, however, these processes are not modeled.

Then there is the scenario where dedicated sub-classes of NonOsModuleInstantiation exist. Here the specific attributes are provided individually per sub-class, e.g. NmInstantiation or LogAndTraceInstantiation.

Those NonOsModuleInstantiations are independent from the startup behavior implementation. If a stack implementation decides to implement a specific functional cluster in a dedicated Process, then the specific NonOsModuleInstantiation will also be part of a ProcessToMachineMapping.

But, if the stack implementation decides to implement a specific functional cluster as a library (or make the functionality part of another functional cluster), then the specific NonOsModuleInstantiation just defines the configuration values for that functionality and does not explicitly take part in a ProcessToMachineMapping.

Another scenario is a rather distributed nature of a functional cluster, where there is no need to provide centralized configuration means. This is applicable for example to PersistencyDeployment or PlatformHealthManagementContribution.

The functional behavior of the functional cluster is determined by the sum of several contributions. There is no single configuration entity provided.

Nevertheless, if a stack implementation decides to implement such a distributed functional cluster as a single Executable, the GenericModuleInstantiation can be used to define the startup behavior for a specific machine.

The configuration settings for individual Adaptive Autosar modules are covered by specializations of the abstract class AdaptiveModuleInstantiation.

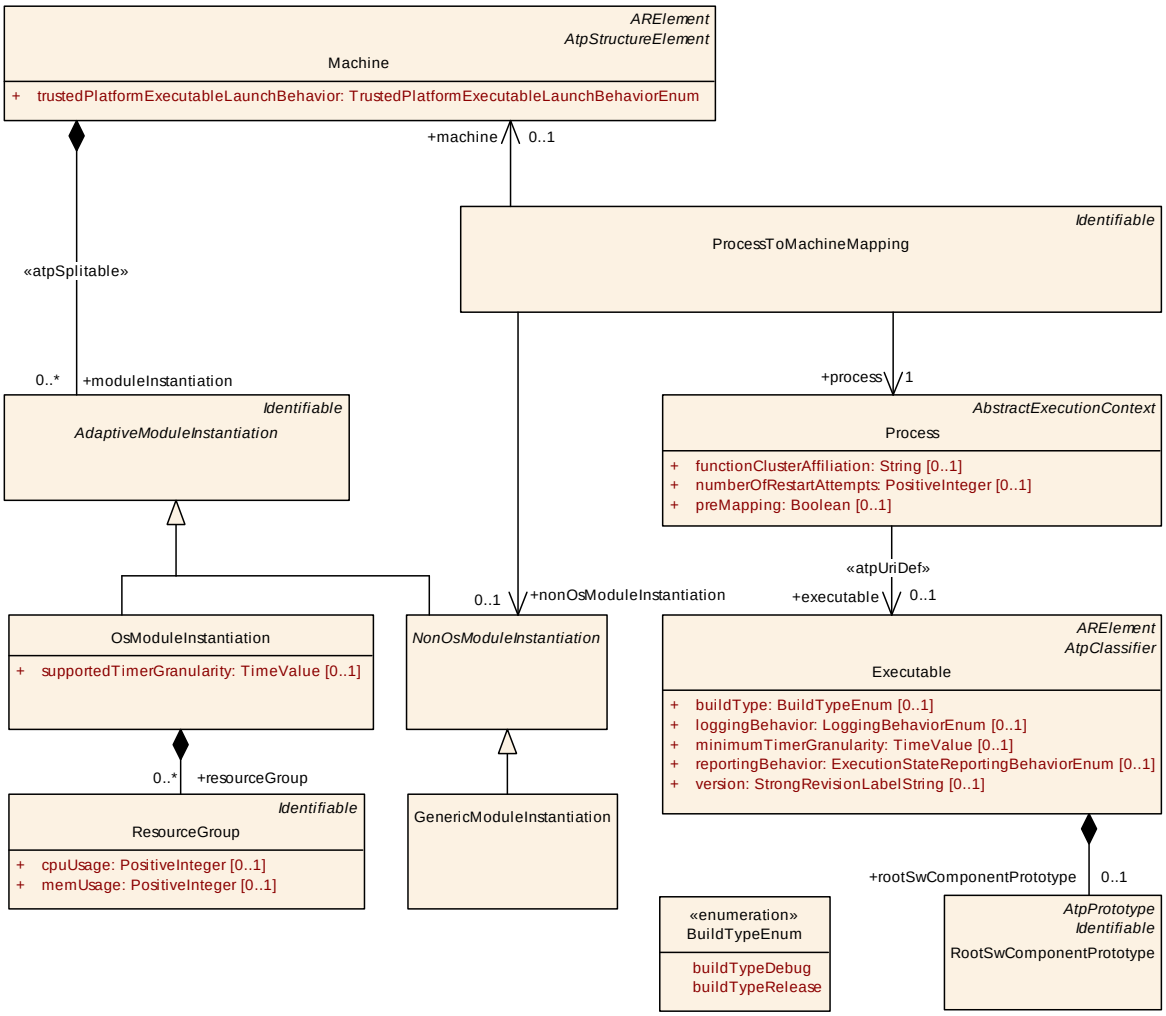


Figure 9.1: Adaptive Autosar Module Configuration

Class	***AdaptiveModuleInstantiation*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModuleImplementation			
Note	This meta-class defines the abstract attributes for the configuration of an adaptive autosar module instance on a specific machine. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*NonOsModuleInstantiation*, OsModuleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 9.1: AdaptiveModuleInstantiation

Each Adaptive Autosar module other than OS can be assigned to a `Process` with the `ProcessToMachineMapping`.

[constr_1490]{DRAFT} **Allowed value for `Executable.category` if `ProcessToMachineMapping` references a `NonOsModuleInstantiation`** ⌈If a `ProcessToMachineMapping` references a `NonOsModuleInstantiation`, then the

Process referenced in the role ProcessToMachineMapping.process shall only refer (in the role Process.executable) to an Executable where attribute Executable.category is set to PLATFORM_LEVEL (see [constr_1605]).⌋()

Please note that the model relation described in [constr_1490] is sketched in Figure 9.1.

The meta-class GenericModuleInstantiation can be used to define configuration settings of generic modules and modules that are not standardized by AUTOSAR. Different modules are distinguishable by the category attribute.

Please note that both elements are Identifiable and therefore are able to describe special data (sdg), by which means it is possible to define generic custom settings that are not represented by the standard model. For more information, please refer to the AUTOSAR Generic Structure Template [6].

[TPS_MANI_03096]{DRAFT} **Machine-specific configuration settings for a generic module** ⌈The Machine-specific configuration settings for a generic module are collected in GenericModuleInstantiation where the value of attribute category value denotes the module.⌋(*RS_MANI_00023*)

Class	**GenericModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class defines the attributes for the generic module configuration on a specific machine. Different modules are distinguishable by the category attribute. This element can also be used to describe modules that are not standardized by AUTOSAR. **Tags:**atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *NonOsModule Instantiation*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 9.2: GenericModuleInstantiation

9.1 OS Module configuration

[TPS_MANI_03098]{DRAFT} **Machine-specific configuration settings for the OS module** ⌈The Machine-specific configuration settings for the OS module are collected in OsModuleInstantiation.⌋(*RS_MANI_00023*)

Class	**OsModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class defines the attributes for the OS configuration on a specific machine. **Tags:**atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
resourceGroup	ResourceGroup	*	aggr	This represents the collection of ResourceGroups owned by the enclosing OsModuleImplementation. **Tags:**atp.Status=draft
supportedTimer Granularity	TimeValue	0..1	attr	This attribute describes the supported timer granularity (TimeValue of one tick). **Tags:**atp.Status=draft

Table 9.3: OsModuleInstantiation

Class	***NonOsModuleInstantiation*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class defines the abstract attributes for the configuration of an adaptive autosar module other than the OS module. **Tags:**atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	CryptoModuleInstantiation, DoIpInstantiation, GenericModuleInstantiation, IamModuleInstantiation, *Ids PlatformInstantiation*, LogAndTraceInstantiation, NmInstantiation, TimeSyncModuleInstantiation, Ucm ModuleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 9.4: NonOsModuleInstantiation

AUTOSAR supports the configuration of `ResourceGroup`s in the `OsModuleInstantiation` of the `Machine` that correspond for example to cgroups (aka control groups) in Linux. `ResourceGroup`s provide a mechanism to manage system resources by partitioning constraints like `cpuUsage` and `memUsage` into groups that limit the resource usage for a collection of processes (see also [TPS_MANI_01017]).

[constr_1661]{DRAFT} **Multiplicity of `OsModuleInstantiation.resourceGroup`** ⌈Any given `OsModuleInstantiation` shall always define at least one `resourceGroup`.⌋*()*

The rationale for [constr_1661] is that the `StateDependentStartupConfig` requires a reference to a `ResourceGroup`.

More information about the semantics of meta-class `ResourceGroup` can be found in [SWS_OSI_02001].

Class	**ResourceGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class represents a resource group that limits the resource usage of a collection of processes. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cpuUsage	PositiveInteger	0..1	attr	CPU resource limit in percentage of the total CPU capacity on the machine.
memUsage	PositiveInteger	0..1	attr	Memory limit in bytes.

Table 9.5: ResourceGroup

9.2 Persistency Deployment

9.2.1 Overview

This chapter explains the part of the support for persistent storage in terms of mapping of concrete storage models to the corresponding parts of the application software.

[TPS_MANI_01205]{DRAFT} **Semantics of meta-class `PersistencyDeployment`** ⌈Abstract meta-class `PersistencyDeployment` provides shared attributes to more specific specializations.⌋ *(RS_MANI_00027)*

Class	***PersistencyDeployment*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This abstract meta-class serves as a base class for concrete classes representing different aspects of persistency. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableExclusivePackageElement*, *UploadablePackageElement*			
Subclasses	PersistencyFileStorage, PersistencyKeyValueStorage			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
maximum AllowedSize	PositiveUnlimitedInteger	0..1	attr	The value of this attribute represents the maximum size allowed at deployment time for the enclosing Persistency Deployment.
minimum SustainedSize	PositiveInteger	0..1	attr	The value of this attribute represents the minimum size guaranteed at deployment time for the enclosing PersistencyDeployment.
redundancy Handling	PersistencyRedundancy Handling	*	aggr	This aggregation represents the chosen approaches to handle redundancy. **Tags:**atp.Status=draft
updateStrategy	PersistencyCollection LevelUpdateStrategy Enum	1	attr	This attribute shall be used to specify the update strategy of the respective PersistencyDeployment as a whole.

Table 9.6: PersistencyDeployment

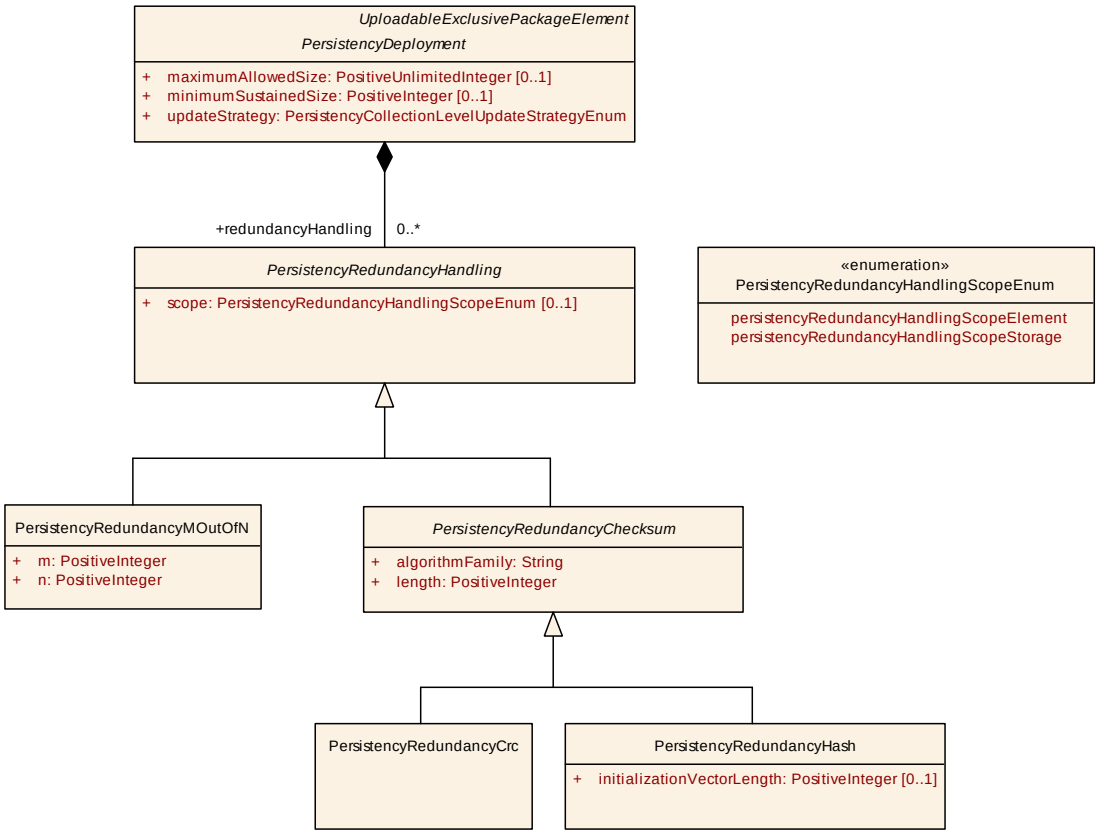


Figure 9.2: Modeling of the abstract base class `PersistencyDeployment`

[TPS_MANI_01321]{DRAFT} **Semantics of meta-class `PersistencyDeploymentElement`** ⌈Meta-class `PersistencyDeploymentElement` represents an abstract base class for the modeling of different aspects of persistency on element level.⌋ *(RS_MANI_00027)*

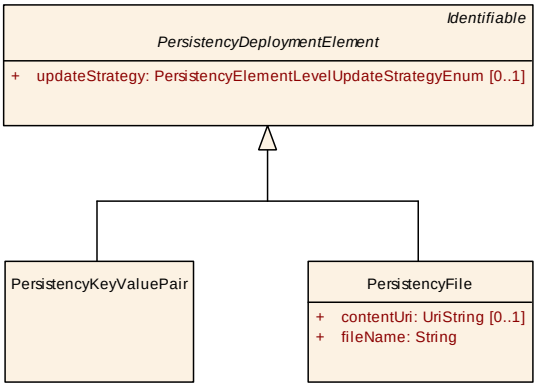


Figure 9.3: Modeling of the abstract base class `PersistencyDeploymentElement`

Class	***PersistencyDeploymentElement*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This abstract meta-class serves as a base class for concrete classes representing different aspects of elements of a PersistencyDeployment. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	PersistencyFile, PersistencyKeyValuePair			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
updateStrategy	PersistencyElement LevelUpdateStrategy Enum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyDeploymentElement.

Table 9.7: PersistencyDeploymentElement

[TPS_MANI_01322]{DRAFT} **Semantics of meta-class `PersistencyPortPrototypeToDeploymentMapping`** ⌈Meta-class `PersistencyPortPrototypeToDeploymentMapping` represents an abstract base class for the modeling of the mapping of concrete persistency cases (key-value storage, file storage) to a `PortPrototype` and a `Process`.⌋*(RS_MANI_00027)*

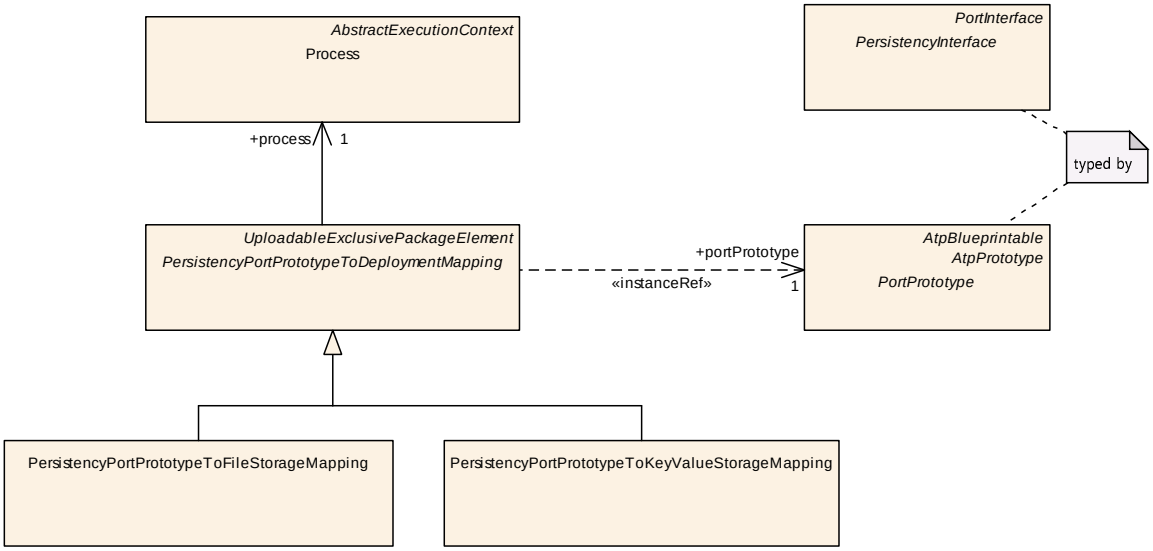


Figure 9.4: Modeling of the abstract base class `PersistencyPortPrototypeToDeploymentMapping`

Class	***PersistencyPortPrototypeToDeploymentMapping*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency
Note	This abstract bas class implements the shared functionality of all mapping between a PortPrototype, a Process, and a specific subclass of PersistencyDeployment. **Tags:**atp.Status=draft
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableExclusivePackageElement*, *UploadablePackageElement*

▽

△

Class	*PersistencyPortPrototypeToDeploymentMapping* (abstract)			
Subclasses	PersistencyPortPrototypeToFileStorageMapping, PersistencyPortPrototypeToKeyValueStorageMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
portPrototype	PortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **Tags:**atp.Status=draft **InstanceRef implemented by:**PortPrototypeIn ExecutableInstanceRef
process	Process	1	ref	This reference represents the process required as context for the mapping. **Tags:**atp.Status=draft

Table 9.8: PersistencyPortPrototypeToDeploymentMapping

9.2.1.1 Redundancy Handling

[TPS_MANI_01206]{DRAFT} **Modeling of redundancy in the context of `PersistencyDeployment`** ⌈The deployment level provides the ability to provide a more detailed definition of redundant behavior for both key-value storage and file storage.

This modeling is attached to the abstract base class `PersistencyDeployment` in order to let both aspects of persistency on the *AUTOSAR adaptive platform* benefit from the existence of meta-class `PersistencyRedundancyHandling`.⌋*(RS_MANI_00027)*

[constr_1710]{DRAFT} **Consistency of values of attributes `PersistencyInterface.redundancy` and `PersistencyRedundancyHandling.scope`** ⌈If attribute `PersistencyInterface.redundancy` is set to value `PersistencyRedundancyEnum.redundantPerElement` then attribute `PersistencyRedundancyHandling.scope` shall be set to `PersistencyRedundancyHandlingScopeEnum.persistencyRedundancyHandlingScopeElement` for at least one `PersistencyRedundancyHandling` aggregated by the corresponding `PersistencyDeployment`.⌋*()*

9.2.1.2 Update Handling

[TPS_MANI_01155]{DRAFT} **`PersistencyDeployment.updateStrategy` overrides `PersistencyInterface.updateStrategy`** ⌈The value of attribute `PersistencyDeployment.updateStrategy` shall overrule the value of `PersistencyInterface.updateStrategy` for any combination of `PersistencyInterface` mapped to a `PersistencyDeployment` by means of a `PersistencyPortPrototypeToDeploymentMapping`.⌋*(RS_MANI_00027)*

[TPS_MANI_01147]{DRAFT} **Semantics of `PersistencyDeployment.updateStrategy`** ⌈The attribute `PersistencyDeployment.updateStrategy` shall be used to specify the strategy for updating the actual persistent elements.

This update strategy shall be applied to the PersistencyDeployment as a whole except for the explicitly modeled PersistencyDeploymentElements that define their own updateStrategy.⌋(*RS_MANI_00027*)

[TPS_MANI_01157]{DRAFT} **Semantics of updateStrategy on collection level** ⌈The semantics of attribute updateStrategy on collection level is specified in Table 9.9.⌋(*RS_MANI_00027*)

updateStrategy	Use Case: Installation	Use Case: Update
delete	irrelevant	delete all elements not contained in current manifest
keepExisting	irrelevant	keep all elements not contained in current manifest

Table 9.9: Semantics of updateStrategy on collection level

[TPS_MANI_01313]{DRAFT} **Definition of updateStrategy on element level** ⌈The definition of the update strategy on element level is modeled by means of the abstract base class PersistencyDeploymentElement (and its attribute updateStrategy) from which the concrete sub-classes for persistency elements are derived.⌋(*RS_MANI_00027*)

[TPS_MANI_01159]{DRAFT} **Semantics of updateStrategy on element level** ⌈The semantics of attribute updateStrategy on element level is specified in Table 9.10.⌋(*RS_MANI_00027*)

updateStrategy	Use Case: Installation	Use Case: Update
delete	don't create	remove
keepExisting	create	do nothing
overwrite	create	replace

Table 9.10: Semantics of updateStrategy on element level

[TPS_MANI_01148]{DRAFT} **Semantics of PersistencyDeploymentElement.updateStrategy** ⌈The attribute PersistencyDeploymentElement.updateStrategy can be used to specify the strategy for updating the actual persistent element that corresponds to PersistencyDeploymentElement.⌋(*RS_MANI_00027*)

[TPS_MANI_01156]{DRAFT} **PersistencyDeploymentElement.updateStrategy overrides PersistencyDeployment.updateStrategy** ⌈The value specified for PersistencyDeploymentElement.updateStrategy overrides the value of PersistencyDeployment.updateStrategy for this specific PersistencyDeploymentElement.⌋(*RS_MANI_00027*)

[TPS_MANI_01182]{DRAFT} **Value of PersistencyDeploymentElement.updateStrategy overrides PersistencyInterfaceElement.updateStrategy** ⌈The value of attribute PersistencyDeploymentElement.updateStrategy overrides the value of attribute PersistencyInterfaceElement.updateStrategy⌋(*RS_MANI_00027*)

This means that the integrator of the software gets the authority to either agree to the designer's point of view or else overrule the designer's decision based on superior knowledge regarding the integration strategy.

9.2.1.3 Size Handling

[TPS_MANI_01196]{DRAFT} **Semantics of `PersistencyDeployment.minimumSustainedSize`** ⌈Attribute `PersistencyDeployment.minimumSustainedSize` can be used for the definition of a **minimum amount of storage** that the `PersistencyDeployment` will need to allocate from an integrator's point of view.

It is the responsibility of the underlying platform to make sure that this minimum amount of storage is available at any time.⌋*(RS_MANI_00027)*

[TPS_MANI_01197]{DRAFT} **Semantics of `PersistencyDeployment.maximumAllowedSize`** ⌈Attribute `PersistencyDeployment.maximumAllowedSize` can be used for the definition of the **maximum amount of storage** that the `PersistencyDeployment` may allocate at runtime from an integrator's point of view.

The existence of `PersistencyDeployment.maximumAllowedSize` does not constitute a binding requirement to the platform that this amount of storage shall be available at any time.⌋*(RS_MANI_00027)*

For explanation, the amount of storage available shall be at least the sum of the values of `minimumSustainedSize`.

That said, it is consequently plausible that storage might be exceeded if more than the minimum amount of storage (let alone the maximum amount) is allocated by all the key-value storage at the same time.

9.2.1.4 Security Handling

The encryption and/or authentication of data stored in a Key-Value Storage or File Storage is described in the manifest by `PersistencyDeploymentToCryptoKeySlotMapping` or `PersistencyDeploymentElementToCryptoKeySlotMapping` that are described in more detail in chapter 12.4 and chapter 12.5.

If the `PersistencyDeploymentToCryptoKeySlotMapping.keySlotUsage` or `PersistencyDeploymentElementToCryptoKeySlotMapping.keySlotUsage` is set to `encryption`, the Persistency cluster shall encrypt the data before storing it to the persistent memory or shall decrypt the data after reading it from persistent memory.

If the `PersistencyDeploymentToCryptoKeySlotMapping.keySlotUsage` or `PersistencyDeploymentElementToCryptoKeySlotMapping.keySlotUsage` is set to `verification`, the Persistency cluster shall sign the data before storing

it to the persistent memory or verify the signature of the data after reading it from persistent memory.

Please note that the `PersistencyDeploymentToCryptoKeySlotMapping` is able to define a `verificationHash` that shall by used by the PersistencyCluster to verify the data. The same is true for the `PersistencyDeploymentElementToCryptoKeySlotMapping.verificationHash`.

9.2.2 Deployment of Persistent Key-Value Storage

[TPS_MANI_01079]{DRAFT} **Semantics of `PersistencyKeyValueStorage`** ⌈Meta-class `PersistencyKeyValueStorage` represents an actual key-value storage used for persistently storing data.⌋ *(RS_MANI_00027)*

Class	**PersistencyKeyValueStorage**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to model a key-value storage on deployment level. **Tags:** atp.Status=draft atp.recommendedPackage=PersistencyKeyValueStorages			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyDeployment*, *Referrable*, *UploadableExclusivePackageElement*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
keyValuePair	PersistencyKeyValuePair	*	aggr	This aggregation represents the key-value-pairs owned by the enclosing PersistencyKeyValueStorage. **Tags:**atp.Status=draft
uri	UriString	0..1	attr	This attribute holds the storage location for the PersistencyKeyValueStorage, e.g. file on the file system.

Table 9.11: PersistencyKeyValueStorage

[TPS_MANI_01144]{DRAFT} **Semantics of `PersistencyKeyValuePair`** ⌈Meta-class `PersistencyKeyValuePair` represents an **entry** to a key-value storage (formalized by `PersistencyKeyValueStorage`) used for persistently storing data.⌋ *(RS_MANI_00027)*

Class	**PersistencyKeyValuePair**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to formally model a key-value pair in the context of the deployment of persistency. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PersistencyDeploymentElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**PersistencyKeyValuePair**			
initValue	ValueSpecification	0..1	aggr	This aggregation represents the ability to define an initial value for the value side of the key-value pair. Please note that it does not make sense to configure an initial value if the PersistencyDeploymentElement.updateStrategy is set to the value delete. **Tags:**atp.Status=draft
valueDataType	AbstractImplementationDataType	1	ref	This reference represents the data type applicable for the value of the key-value pair. **Tags:**atp.Status=draft

Table 9.12: PersistencyKeyValuePair

The modeling of PersistencyKeyValuePair aggregated in the role PersistencyKeyValueStorage.keyValuePair is optional. It would be possible to use persistency functionality regardless of the existence of keyValuePair.

However, the presence of keyValuePair gives more freedom and ways for the customization of behavior.

[TPS_MANI_01078]{DRAFT} **Semantics of PersistencyPortPrototypeToKeyValueStorageMapping** ⌈Meta-class PersistencyPortPrototypeToKeyValueStorageMapping has the ability to map a specific PortPrototype referenced in the role portPrototype to a PersistencyKeyValueStorage referenced in the role keyValueStorage.

The mapping also comprises a reference to meta-class process in order to accommodate for the fact that identical combinations of keyValueStorage and portPrototype may or may not apply for a given Process that represents the enclosing Executable at runtime.⌋*(RS_MANI_00027)*

[constr_1555]{DRAFT} **Restriction applicable for PersistencyPortPrototypeToKeyValueStorageMapping.portPrototype** ⌈The reference PersistencyPortPrototypeToKeyValueStorageMapping.portPrototype shall only be used for a PortPrototype typed by a PersistencyKeyValueStorageInterface.⌋*()*

Class	**PersistencyPortPrototypeToKeyValueStorageMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency
Note	This meta-class represents the ability to define a mapping between a PortPrototype and a key-value storage. **Tags:** atp.Status=draft atp.recommendedPackage=PersistencyPortPrototypeToKeyValueStorageMappings
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PersistencyPortPrototypeToDeploymentMapping*, *Referrable*, *UploadableExclusivePackage Element*, *UploadablePackageElement*

▽

△

Class	**PersistencyPortPrototypeToKeyValueStorageMapping**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
keyValue Storage	PersistencyKeyValue Storage	1	ref	This reference represents the mapped key-value storage. **Tags:**atp.Status=draft

Table 9.13: PersistencyPortPrototypeToKeyValueStorageMapping

Please note that typically the existence of `PersistencyKeyValueStorage.keyValuePair` depends on the existence of `PersistencyKeyValueStorageInterface.dataElement`.

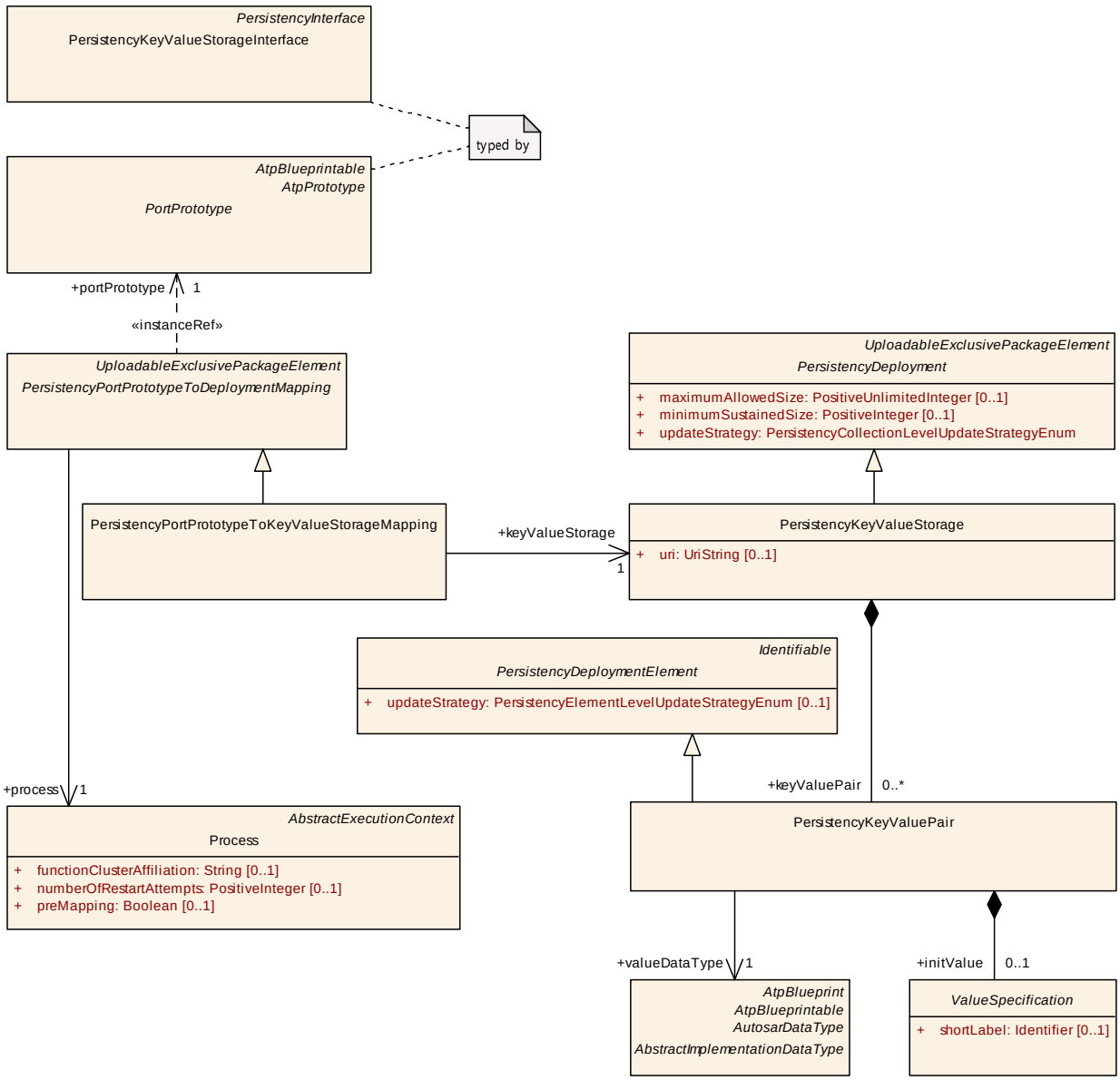


Figure 9.5: Connect a specific `PortPrototype` to a `PersistencyKeyValueStorage`

On the other hand, if a PersistencyKeyValueStorage contains PersistencyKeyValuePairs that do not correspond to any dataElements of the PersistencyKeyValueStorageInterface that is mapped (indirectly) via PersistencyPortPrototypeToKeyValueStorageMapping then those keyValuePairs are created within the PersistencyKeyValueStorage.

[TPS_MANI_01146]{DRAFT} **Initial value for PersistencyKeyValuePair** ⌈It is possible to define an initial value for a given PersistencyKeyValuePair by means of the aggregation of ValueSpecification in the role initValue.⌋*(RS_MANI_00027)*

[constr_1554]{DRAFT} **Restriction regarding PersistencyKeyValuePair.initValue** ⌈The concrete sub-class of ValueSpecification aggregated in the role PersistencyKeyValuePair.initValue shall not (after resolving a possible redirection by means of ConstantReference) be one of the following:

- ApplicationValueSpecification
- ApplicationRuleBasedValueSpecification
- ReferenceValueSpecification

⌋*()*

[TPS_MANI_01315]{DRAFT} **PersistencyKeyValuePair.initValue overrides PersistencyDataRequiredComSpec.initValue** ⌈The value of attribute PersistencyKeyValuePair.initValue shall overrule the value of PersistencyDataRequiredComSpec.initValue for any combination of PersistencyKeyValueStorageInterface mapped to a PersistencyKeyValueStorage by means of a PersistencyPortPrototypeToKeyValueStorageMapping.⌋*(RS_MANI_00027)*

This means that the integrator of the software gets the authority to either agree to the designer's point of view or else overrule the designer's decision based on superior knowledge regarding the integration strategy.

[constr_1582]{DRAFT} **PersistencyKeyValuePair.valueDataType shall match to AbstractImplementationDataType for the corresponding PersistencyDataElement** ⌈Each PersistencyKeyValuePair.valueDataType shall match the AbstractImplementationDataType that either directly or indirectly (via the applicable DataTypeMap) types the corresponding (based on identical values of the respective shortName) PersistencyDataElement.⌋*()*

[constr_1666]{DRAFT} **References from PersistencyPortPrototypeToKeyValueStorageMapping to PersistencyKeyValueStorage** ⌈Each PersistencyKeyValueStorage shall only be referenced by at most one PersistencyPortPrototypeToKeyValueStorageMapping.⌋*()*

[TPS_MANI_01323]{DRAFT} **Matching pairs of PersistencyDataElement and PersistencyKeyValuePair** ⌈Matching pairs of PersistencyDataElement and

PersistencyKeyValuePair shall be identified by having the identical value of attribute shortName within the scope of a PersistencyKeyValueStorageInterface (or a PortPrototype typed by the PersistencyKeyValueStorageInterface) mapped to a PersistencyKeyValueStorage by means of a PersistencyPortPrototypeToKeyValueStorageMapping.⌋ *(RS_MANI_00027)*

9.2.3 Deployment of File Storage

[TPS_MANI_01150]{DRAFT} **Semantics of PersistencyFileStorage** ⌈A PortPrototype typed by a PersistencyFileStorageInterface actually builds an abstraction for an entire directory of files.

This abstraction is also visible in the deployment by means of the existence of the companion meta-class PersistencyFileStorage.

This approach allows for the dynamic creation and/or deletion of files during runtime while still keeping the structural model of the file interaction static.⌋ *(RS_MANI_00027)*

Class	**PersistencyFileStorage**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class comes with the ability to define a collection of single files (directory) that creates the deployment-side counterpart to a PortPrototype typed by a PersistencyFileStorageInterface. **Tags:** atp.Status=draft atp.recommendedPackage=PersistencyFileStorages			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyDeployment*, *Referrable*, *UploadableExclusivePackageElement*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
file	PersistencyFile	*	aggr	This aggregation represents the collection of files aggregated by the PersistencyFileStorage. **Tags:**atp.Status=draft
uri	UriString	1	attr	This attribute holds the storage location for the PersistencyFileStorage, e.g. a directory on the file system.

Table 9.14: PersistencyFileStorage

At one point, however, it is necessary to boil down the relation of such a PortPrototype typed by a PersistencyFileStorageInterface to individual files and how these individual files are represented on the file system themselves.

This aspect is covered by the modeling of meta-class PersistencyPortPrototypeToFileStorageMapping, as depicted in Figure 9.6.

[TPS_MANI_01080]{DRAFT} **Semantics of meta-class PersistencyPortPrototypeToFileStorageMapping** ⌈Meta-class PersistencyPortPrototypeToFileStorageMapping creates a mapping between a PortPrototype referenced in the role portPrototype to a PersistencyFileStorage referenced

in the role `fileStorage` under consideration of a `Process` referenced in the role `process`.⌋*(RS_MANI_00027)*

Class	**PersistencyPortPrototypeToFileStorageMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to define a mapping between a collection of files on deployment level to a given PortPrototype. **Tags:** atp.Status=draft atp.recommendedPackage=PersistencyPortPrototypeToFileStorageMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PersistencyPortPrototypeToDeploymentMapping*, *Referrable*, *UploadableExclusivePackage Element*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
fileStorage	PersistencyFileStorage	1	ref	This reference represents the mapped file storage. **Tags:**atp.Status=draft

Table 9.15: PersistencyPortPrototypeToFileStorageMapping

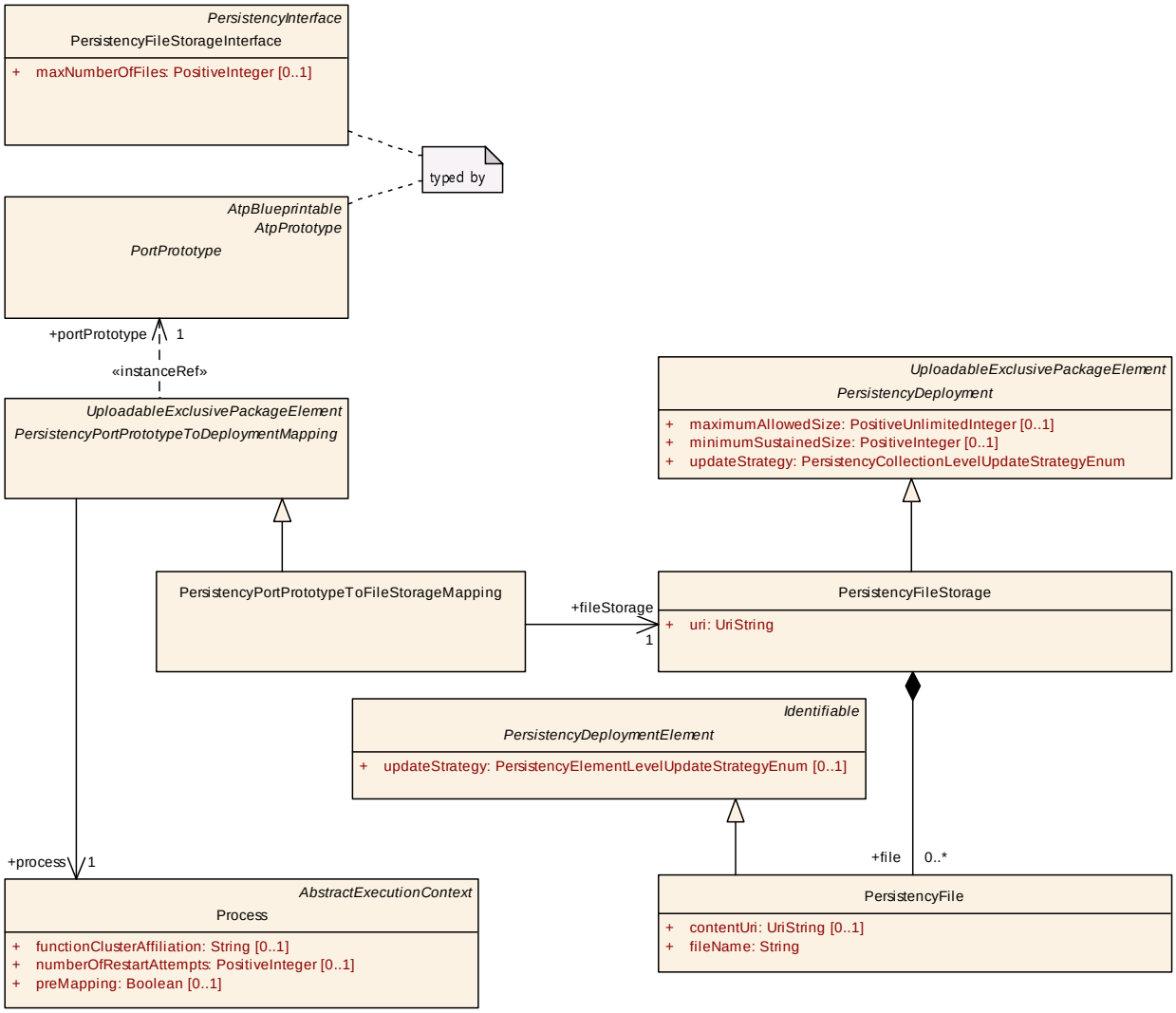


Figure 9.6: Connect a specific `PortPrototype` to a `PersistencyFile`

[TPS_MANI_01149]{DRAFT} **Semantics of `PersistencyFileStorage.file`** ⌈The usage of `PersistencyFileStorage.file` allows for the explicit modeling of elements of the `PersistencyFileStorage`.

The creation of this aggregation is optional. It can be used to define the update strategy and/or initial content of selected files.⌋*(RS_MANI_00027)*

[constr_1556]{DRAFT} **Restriction applicable for `PersistencyPortPrototypeToFileStorageMapping.portPrototype`** ⌈The reference `PersistencyPortPrototypeToFileStorageMapping.portPrototype` shall only be used for a `PortPrototype` typed by a `PersistencyFileStorageInterface`.⌋*()*

Class	**PersistencyFile**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the model of a file as part of the persistency on deployment level. **Tags:** atp.Status=draft atp.recommendedPackage=PersistencyFiles			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PersistencyDeploymentElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contentUri	UriString	0..1	attr	This attribute represents the URI that identifies the initial content of the PersistencyFile.
fileName	String	1	attr	This attribute holds filename part of the storage location for the PersistencyFile, e.g. file on the file system. **Tags:**atp.Status=draft

Table 9.16: PersistencyFile

[TPS_MANI_01179]{DRAFT} **Semantics of `PersistencyFileElement.contentUri`/`PersistencyFile.contentUri` vs. `PersistencyFileStorage.uri` and `PersistencyFileElement.fileName`/`PersistencyFile.fileName`** ⌈Attributes `PersistencyFileElement.contentUri` and (after deployment) `PersistencyFile.contentUri` describe the URI of the file storage that is used to initialize the `PersistencyFile` (used during install or update).

On the other hand, the combination of `PersistencyFileStorage.uri` and the `PersistencyFileElement.fileName` or (after deployment) `PersistencyFile.fileName` denote the position of the `PersistencyFile` in the ECU (used at runtime).⌋*(RS_MANI_00027)*

[constr_1589]{DRAFT} **Value of `file.fileName`** ⌈Within the scope of any given `PersistencyFileStorage`, the value of all `file.fileName` shall be unique.

A `fileName` is considered unique if there are no other `fileName`s with **exactly** the same sequence of characters[1].⌋*()*

[TPS_MANI_01187]{DRAFT} **Matching pairs of `PersistencyFileElement` and `PersistencyFile`** ⌈Matching pairs of `PersistencyFileElement` and `PersistencyFile` shall be identified by having the identical value of attribute `shortName` within the scope of a `PersistencyFileStorageInterface` (or a `PortPrototype`

[1]The characters “x” and “X” are not considered as identical characters for this purpose.

typed by the PersistencyFileStorageInterface) mapped to a PersistencyFileStorage by means of a PersistencyPortPrototypeToFileStorageMapping.⌋(*RS_MANI_00027*)

[constr_1613]{DRAFT} **File name of matching pairs of PersistencyFileElement and PersistencyFile** ⌈The value of attributes PersistencyFileElement.fileName and PersistencyFile.fileName shall be identical for matching pairs (as identified by the application of [TPS_MANI_01187]) of PersistencyFileStorage and PersistencyFile.⌋()

[constr_1667]{DRAFT} **References from PersistencyPortPrototypeToFileStorageMapping to PersistencyFileStorage** ⌈Each PersistencyFileStorage shall only be referenced by at most one PersistencyPortPrototypeToFileStorageMapping.⌋()

9.3 Platform Health Management Deployment

9.3.1 Overview

This chapter explains the interaction of application software with the Platform Health Management [12].

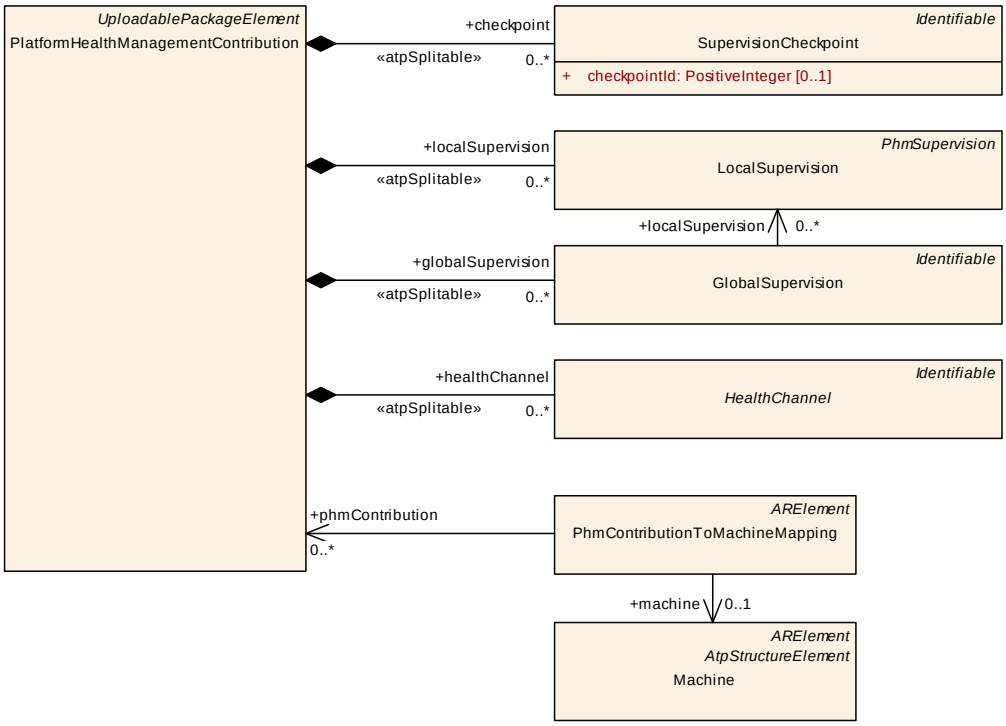


Figure 9.7: Modeling of PlatformHealthManagementContribution

The PlatformHealthManagementContribution allows describing aspects for the deployment of configuration how the Platform Health Management shall behave during runtime.

[TPS_MANI_03544]{DRAFT} **Definition of `PlatformHealthManagementContribution`** ⌈The meta-class `PlatformHealthManagementContribution` allows to define a set of configuration entities for the Platform Health Management.⌋*(RS_MANI_00023, RS_MANI_00032)*

The `PlatformHealthManagementContribution` is structured into several aspects which will be described in the following sections:

- Supervision (section 9.3.3)
- Health channels (section 9.3.5)
- Recovery Notification (section 9.3.6)

Class	**PlatformHealthManagementContribution**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a contribution to the Platform Health Management. **Tags:** atp.Status=draft atp.recommendedPackage=PlatformHealthManagementContributions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checkpoint	SupervisionCheckpoint	*	aggr	Collection of checkpoints in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=checkpoint.shortName atp.Status=draft xml.sequenceOffset=10
global Supervision	GlobalSupervision	*	aggr	Collection of GlobalSupervisions in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=globalSupervision.shortName atp.Status=draft xml.sequenceOffset=30
healthChannel	HealthChannel	*	aggr	Collection of HealthChannels in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=healthChannel.shortName atp.Status=draft xml.sequenceOffset=40
local Supervision	LocalSupervision	*	aggr	Collection of LocalSupervisions in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=localSupervision.shortName atp.Status=draft xml.sequenceOffset=20

Table 9.17: PlatformHealthManagementContribution

[TPS_MANI_03502]{DRAFT} **Enabling of `PlatformHealthManagementContribution` on a `Machine`** ⌈To enable an instance of `PlatformHealthManagementContribution` on a specific `Machine` the `PlatformHealthManagementContribution` shall be mapped to the `Machine` via a `PhmContributionToMachineMapping`.⌋*(RS_MANI_00023, RS_MANI_00032)*

[constr_3568]{DRAFT} **No support for cross `PlatformHealthManagementContribution` references** ⌈All references originating on elements aggregated by one `PlatformHealthManagementContribution` shall only refer to elements that are part of the same `PlatformHealthManagementContribution` aggregation chain.⌋ *()*

Class	**PhmContributionToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element associates one or more PlatformHealthManagementContributions with a Machine. **Tags:** atp.Status=draft atp.recommendedPackage=PhmContributionToMachineMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
machine	Machine	0..1	ref	This reference identifies the Machine in the context of the PhmContributionToMachineMapping. **Tags:**atp.Status=draft
phm Contribution	PlatformHealth Management Contribution	*	ref	This reference identifies one or more PlatformHealth ManagementContributions in the context of a Phm ContributionToMachineMapping. **Tags:**atp.Status=draft

Table 9.18: PhmContributionToMachineMapping

An application software can define the usage of several Platform Health Management supervisions (see chapter 3.10.2) and health channels (see chapter 3.10.3).

In order to define the interaction between the application software and the Platform Health Management the `PlatformHealthManagementContribution` creates its own representations of the `RPortPrototype`s typed by the `PhmSupervisedEntityInterface` and `PhmHealthChannelInterface` and creates relations to the application software `RPortPrototype`s (see figure 9.8).

In chapter 3.10.2 it is explained that the application software just calls methods in the context of the respective `RPortPrototype`s to interact with the Platform Health Management. From the application developer these methods have no addressing information, because the identity of the `RPortPrototype` is the identification in the scope of the application software.

The deployed structure (according to figure 9.7) however requires more information when an `API` at the Platform Health Manager is called, namely:

- `RPortPrototype.shortName` i.e. `InstanceSpecifier`
- `Process` identification during runtime.

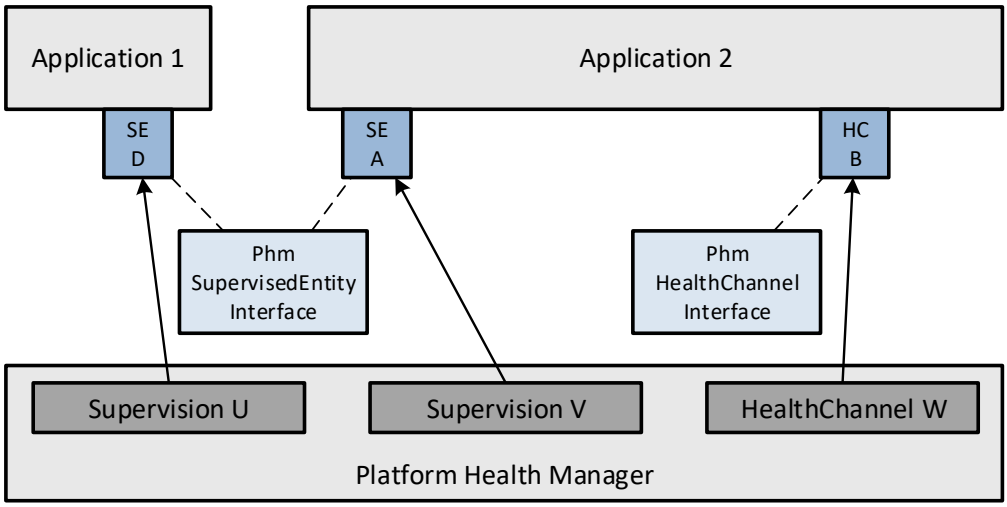


Figure 9.8: Interaction of application software with the platform health manager

These additional arguments have to be injected to the `API` by the implementation of the interaction between the software component and the Platform Health Management (which implements the relations from figure 9.7). The order of this argument injection is determined by the specification of the Platform Health Management APIs.

9.3.2 Relation between design and deployment

The application design in chapters 3.10.2 and 3.10.3 uses the declarations provided in the Interface definitions for `PhmSupervisedEntityInterface` and `PhmHealthChannelInterface`. Specifically the handling of ids (`checkpointId` and `statusId`) requires a synchronized usage with the respective Interface definition.

It is required to establish a contract between the application code and the Phm deployment. The application code shall only use such id values which are declared at the respective Interface definitions (see [TPS_MANI_03623] and [TPS_MANI_03624]).

During the configuration of the Phm the Phm artifacts are created. At deployment state there is no access to the design model available, thus the numeric values used for `checkpointId` and `statusId` at the Interface definition (`PhmSupervisedEntity-Interface` and `PhmHealthChannelInterface`) are not available to the Phm.

Therefore the numeric values for `checkpointId` and `statusId` are replicated in the deployment model of the Phm:

- `SupervisionCheckpoint.checkpointId` replicates `PhmCheckpoint.checkpointId`

- `HealthChannelExternalReportedStatus.statusId` replicates `PhmHealthChannelStatus.statusId`

It is a methodological task to make sure that the Ids correspond to each other and match in value. This consistency can be checked using specific tooling on the deployment model and the design model (see also [TPS_MANI_03623] and [TPS_MANI_03624]).

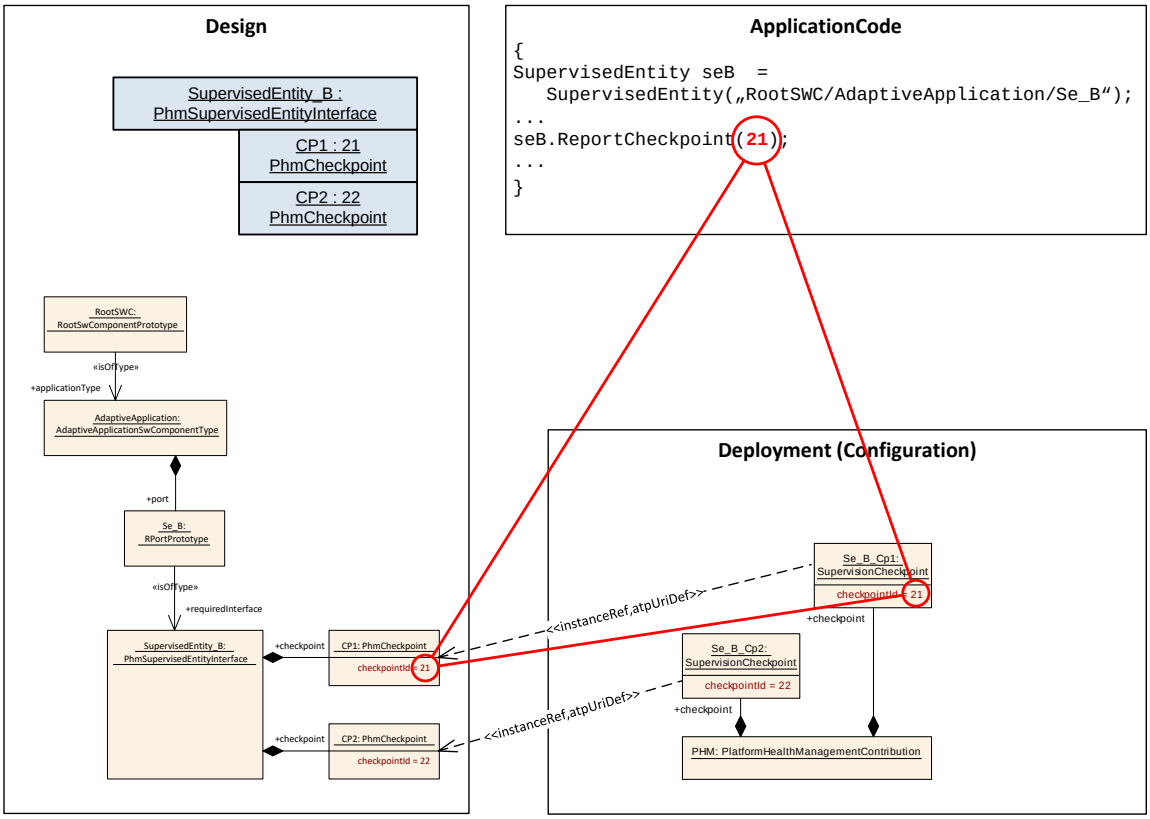


Figure 9.9: Example how IDs have to be in sync

The example in figure 9.9 illustrates that the used IDs have to in sync in order to let design, code, and configuration work together.

9.3.3 Supervision deployment

In the application design chapter of this document the declaration of supervised entities and checkpoints has been described (see section 3.10.2). These declarations provide the view on supervision from the application software code point. Since the application `Executable` can be started multiple times (via individual `Process`es), the configuration of the Platform Health Management needs to cope with these individual `Executable` instances.

[TPS_MANI_03503]{DRAFT} **Applicability of checkpoints to a specific `Process`** ⌈The reference `SupervisionCheckpoint.process` defines to which specific `Process` this `SupervisionCheckpoint` definition shall be applied to.⌋*(RS_MANI_00023, RS_MANI_00032)*

This means that only if a `PhmCheckpoint` is reported from the context of this `Process` it is considered to be this `SupervisionCheckpoint`.

For the Platform Health Management supervision to take effect it is required to define the instance the application is executed in, thus the reference to a `Process` has to be taken into account. In the model the `Process` also defines under which conditions (`StateDependentStartupConfig`) and with which arguments (`ProcessArgument`) the `Executable` will be started.

For the configuration of the Platform Health Management the definition of `SupervisionCheckpoint` is used to stand in for the corresponding `PhmCheckpoint` including the execution context of the respective `Process`.

The attribute `SupervisionCheckpoint.phmCheckpoint` replicates the value of the referenced `PhmCheckpoint.checkpointId`. During deployment the `PhmSupervisedEntityInterface` and its content is no longer available and therefore needs to be made available to the Phm.

[TPS_MANI_03626]{DRAFT} **Consistency of `SupervisionCheckpoint.phmCheckpoint` and `PhmCheckpoint.checkpointId`** ⌈The value of `SupervisionCheckpoint.phmCheckpoint` shall be identical to the value of `PhmCheckpoint.checkpointId` which is referenced in `SupervisionCheckpoint.phmCheckpoint`.⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03505]{DRAFT} **Existence of `SupervisionCheckpoint`** ⌈For each `PhmCheckpoint` in the scope of a `RPortPrototype` typed by a `PhmSupervisedEntityInterface` in the application definition there may be a `SupervisionCheckpoint` defined. The correspondence of the two is defined by the instance reference `SupervisionCheckpoint.phmCheckpoint`⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03506]{DRAFT} **Optionality of `SupervisionCheckpoint`** ⌈It is not required that every `PhmSupervisedEntityInterface` or `PhmCheckpoint` used in the context of the application definition eventually has a corresponding `SupervisionCheckpoint` defined. There may be cases where the application software reports some checkpoints, but they are not considered for a specific supervision.⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03515]{DRAFT} **Expiration tolerance for `LocalSupervision`** ⌈The attribute `LocalSupervision.failedSupervisionCyclesTolerance` defines how many supervision cycles an incorrect supervision is maintained in the state *failed* before it is considered *expired*.⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**LocalSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a LocalSupervision in the context of platform health management contribution. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmSupervision*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
alive Supervision	AliveSupervision	*	aggr	Collection of AliveSupervisions in the context of this Local Supervision. **Tags:**atp.Status=draft
deadline Supervision	DeadlineSupervision	*	aggr	Collection of DeadlineSupervisions in the context of this LocalSupervision. **Tags:**atp.Status=draft
failed Supervision Cycles Tolerance	PositiveInteger	0..1	attr	Defines the acceptable amount of cycles with FAILED supervision status of this LocalSupervision before it is considered EXPIRED. **Tags:**atp.Status=draft
logical Supervision	LogicalSupervision	*	aggr	Collection of LogicalSupervisions in the context of this LocalSupervision. **Tags:**atp.Status=draft
transition	CheckpointTransition	*	aggr	Collection of CheckpointTransitions in the context of this LocalSupervision. **Tags:**atp.Status=draft

Table 9.19: LocalSupervision

Class	**SupervisionCheckpoint**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element contains an instance reference to a RPortPrototype representing a checkpoint for Platform Health Management. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checkpointId	PositiveInteger	0..1	attr	Defines the numeric value which is used to identify the reporting of this SupervisionCheckpoint to the Phm.
phmCheckpoint	PhmCheckpoint	0..1	iref	Instance reference to the PhmCheckpoint defined in the context of a PortInterface. **Tags:**atp.Status=draft **InstanceRef implemented by:**PhmCheckpointIn ExecutableInstanceRef
process	Process	0..1	ref	Reference to the Process this checkoint shall be monitored. **Tags:**atp.Status=draft

Table 9.20: SupervisionCheckpoint

[constr_1764]{DRAFT} **Counterpart of `PhmCheckpoint`** ⌈Each `PhmCheckpoint` shall be referenced once and only once in the role `targetPhmCheckpoint` by a `PhmCheckpointInExecutableInstanceRef` that is aggregated by a `SupervisionCheckpoint`. This reference shall exist **at the time when manifest creation is finished.**⌋*()*

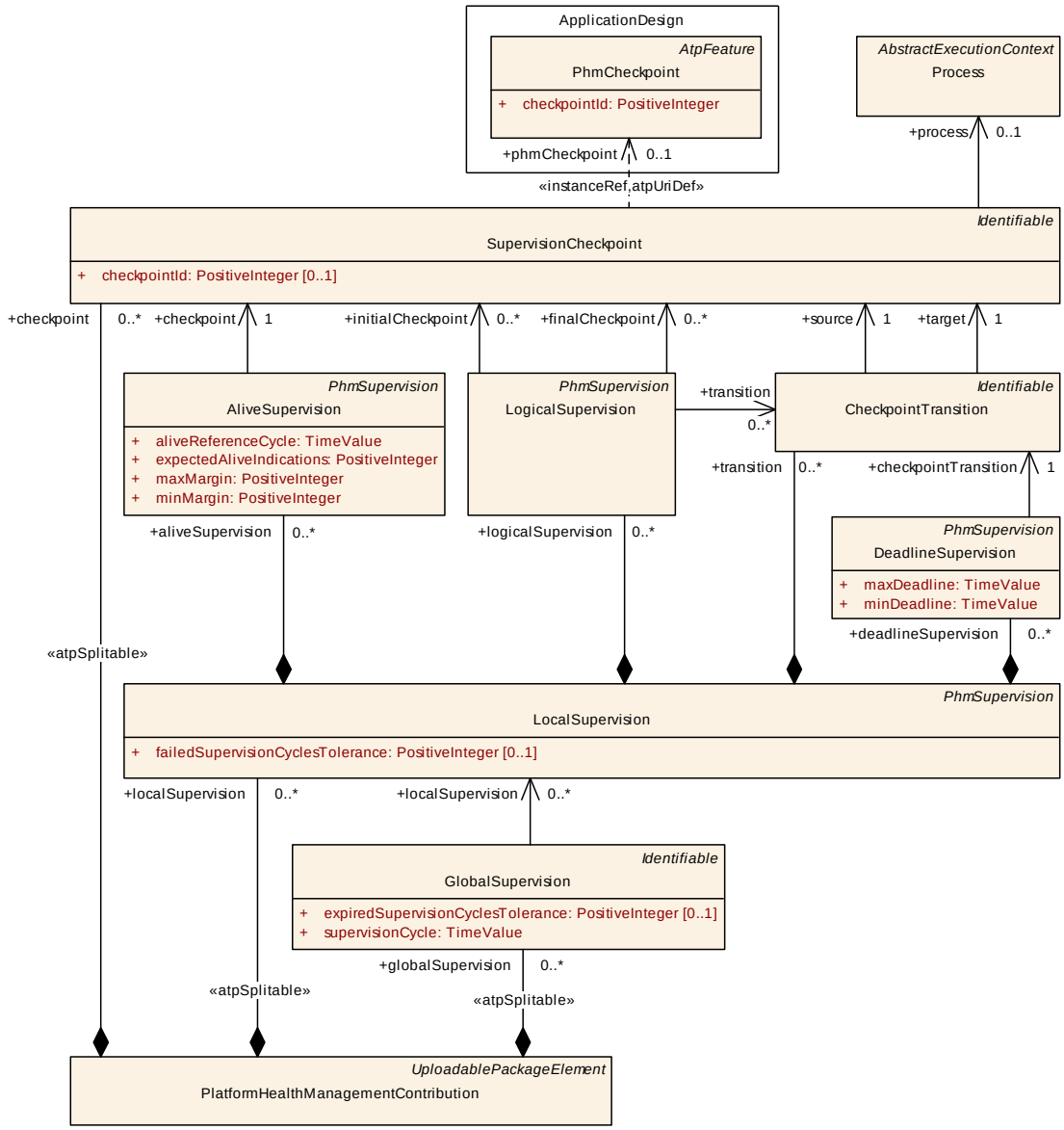


Figure 9.10: Modeling of `LocalSupervision`

Please note that the detailed modeling of the ≪InstanceRef≫ from `SupervisionCheckpoint` to `PhmCheckpoint` is documented in section C.8.

[constr_1739]{DRAFT} **Multiplicity of aggregation `LocalSupervision.transition`** ⌈At the time of deployment of a `LocalSupervision`, at least one aggregation of meta-class `CheckpointTransition` in the role `LocalSupervision.transition` shall exist **if at least one the following conditions** is fulfilled:

- At least one aggregation of `LogicalSupervision` in the role `LocalSupervision.logicalSupervision` exists.
- At least one aggregation of `DeadlineSupervision` in the role `LocalSupervision.deadlineSupervision` exists.

⌋()

[constr_1742]{DRAFT} **Multiplicity of reference `SupervisionCheckpoint.phmCheckpoint`** ⌈At the time of deployment of a `SupervisionCheckpoint`, one reference to meta-class `PhmCheckpoint` in the role `phmCheckpoint` shall exist.⌋*()*

9.3.3.1 `AliveSupervision` definition

In the scope of a `LocalSupervision` an `AliveSupervision` can be defined for a specific `SupervisionCheckpoint`. `LocalSupervision` can be used to define in which timing boundaries one specific checkpoint shall be monitored.

[TPS_MANI_03508]{DRAFT} **Definition of an `AliveSupervision` for a `SupervisionCheckpoint`** ⌈An `AliveSupervision` definition provides attributes to configure the supervision of the referenced `SupervisionCheckpoint`.

- `aliveReferenceCycle` defines the time base used to monitor the reporting of this specific `SupervisionCheckpoint`
- `expectedAliveIndications` defines the number of indications which shall be observed during the time period defined by `aliveReferenceCycle`
- `minMargin` and `maxMargin` define the acceptable deviation from the `expectedAliveIndications` within the time period defined by `aliveReferenceCycle`

⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03575]{DRAFT} **Definition of no minimum alive supervision** ⌈If the value `AliveSupervision.minMargin` equals 0, this defines that no minimum alive supervision shall be performed.⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03576]{DRAFT} **Definition of no maximum alive supervision** ⌈If the value `AliveSupervision.maxMargin` equals INF, this defines that no maximum alive supervision shall be performed.⌋*(RS_MANI_00023, RS_MANI_00032)*

[constr_3539]{DRAFT} **Only one `AliveSupervision` per `SupervisionCheckpoint`** ⌈A `SupervisionCheckpoint` shall only be referenced up to once by an `AliveSupervision` in the role `checkpoint`.⌋*()*

Class	**AliveSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Defines an AliveSupervision for one checkpoint. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmSupervision*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**AliveSupervision**			
aliveReference Cycle	TimeValue	1	attr	Time period at which the Alive Supervision mechanism compares the amount of received Alive Indications for the SupervisionCheckpoint against the expectedAlive Indications. **Tags:**atp.Status=draft
checkpoint	SupervisionCheckpoint	1	ref	Reference to a checkpoint in the context of Alive Supervision. **Tags:**atp.Status=draft
expectedAlive Indications	PositiveInteger	1	attr	Defines the amount of expected Alive Indications of the SupervisionCheckpoint within the aliveReferenceCycle. **Tags:**atp.Status=draft
maxMargin	PositiveInteger	1	attr	Defines the amount of Alive Indications of the Supervision Checkpoint that are acceptable to be additional to the expectedAliveIndications within the aliveReferenceCycle. **Tags:**atp.Status=draft
minMargin	PositiveInteger	1	attr	Defines the amount of Alive Indications of the Supervision Checkpoint that are acceptable to be missing to the expectedAliveIndications within the aliveReferenceCycle. **Tags:**atp.Status=draft

Table 9.21: AliveSupervision

9.3.3.2 `CheckpointTransition` definition

For the definition of further supervision strategies the need to first define possible `CheckpointTransition`s between `SupervisionCheckpoint`s arises. Since the application software design does not provide any transition definition between checkpoints, it is essential to define possible `CheckpointTransition`s.

The definition of `CheckpointTransition`s is done in the scope of the `LocalSupervision` and can be used by the `LogicalSupervision` and `DeadlineSupervision`.

[TPS_MANI_03509]{DRAFT} **Definition of a `CheckpointTransition`** ⌈A `CheckpointTransition` defines one possible transition from the `source SupervisionCheckpoint` to the `target SupervisionCheckpoint`.⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**CheckpointTransition**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Defines one transition between two checkpoints. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	CheckpointTransition			
source	SupervisionCheckpoint	1	ref	Reference to the source checkpoint for this transition. **Tags:**atp.Status=draft
target	SupervisionCheckpoint	1	ref	Reference to the target checkpoint for this transition. **Tags:**atp.Status=draft

Table 9.22: CheckpointTransition

9.3.3.3 `LogicalSupervision` definition

The `LogicalSupervision` defines a supervision graph of allowed `CheckpointTransition`s which is monitored by the Platform Health Management without any timing considerations, just the order of reported checkpoints is considered for the monitoring.

[constr_3540]{DRAFT} **`SupervisionCheckpoint` in supervision graph** ⌈Each `SupervisionCheckpoint` shall only be part of one supervision graph.⌋*()*

When a `SupervisionCheckpoint` belonging to the supervision graph is reported to the Platform Health Management where there is no `CheckpointTransition` defined from the last reported `SupervisionCheckpoint` as `source` to the current reported `SupervisionCheckpoint` as `target`, this situation violates the `LogicalSupervision`.

[TPS_MANI_03510]{DRAFT} **Definition of `LogicalSupervision`** ⌈A `LogicalSupervision` defines relations between `SupervisionCheckpoint`s which form a directed graph from one or more `initialCheckpoint SupervisionCheckpoint`s through a set of `CheckpointTransition`s defined by collection of `transition`s to one or more `finalCheckpoint SupervisionCheckpoint`s.⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	LogicalSupervision			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Defines a LogicalSupervision graph consisting of transitions, initial- and final checkpoints. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmSupervision*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
finalCheckpoint	SupervisionCheckpoint	*	ref	Reference to the final Checkpoint(s) for this Logical Supervision. **Tags:** atp.Status=draft xml.sequenceOffset=20

▽

△

Class	LogicalSupervision			
initialCheckpoint	SupervisionCheckpoint	*	ref	Reference to the initial Checkpoint(s) for this Logical Supervision. **Tags:** atp.Status=draft xml.sequenceOffset=10
transition	CheckpointTransition	*	ref	Reference to the transitions for this LogicalSupervision. **Tags:** atp.Status=draft xml.sequenceOffset=30

Table 9.23: LogicalSupervision

[constr_1736]{DRAFT} **Multiplicity of reference `LogicalSupervision.initialCheckpoint`** ⌈At the time of deployment of a `LogicalSupervision`, at least one reference to meta-class `SupervisionCheckpoint` in the role `initialCheckpoint` shall exist.⌋*()*

[constr_1737]{DRAFT} **Multiplicity of reference `LogicalSupervision.finalCheckpoint`** ⌈At the time of deployment of a `LogicalSupervision`, at least one reference to meta-class `SupervisionCheckpoint` in the role `finalCheckpoint` shall exist.⌋*()*

[constr_1740]{DRAFT} **Multiplicity of reference `LogicalSupervision.transition`** ⌈At the time of deployment of a `LogicalSupervision`, at least one reference to meta-class `CheckpointTransition` in the role `LogicalSupervision.transition` shall exist.⌋*()*

9.3.3.4 `DeadlineSupervision` definition

The `DeadlineSupervision` defines timing attributes for one specific `CheckpointTransition`.

[TPS_MANI_03511]{DRAFT} **Definition of `DeadlineSupervision`** ⌈A `DeadlineSupervision` defines timing attributes which are monitored by the Platform Health Management for one specific `CheckpointTransition`.⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03573]{DRAFT} **Definition of no minimum deadline supervision** ⌈If the value `DeadlineSupervision.minDeadline` equals 0, this defines that no minimum deadline supervision shall be performed.⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03574]{DRAFT} **Definition of no maximum deadline supervision** ⌈If the value `DeadlineSupervision.maxDeadline` equals INF, this defines that no maximum deadline supervision shall be performed.⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**DeadlineSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Defines an DeadlineSupervision for one transition. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmSupervision*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checkpoint Transition	CheckpointTransition	1	ref	Reference to the transition in the context of a Deadline Supervision. **Tags:**atp.Status=draft
maxDeadline	TimeValue	1	attr	Defines the longest time span before which the deadline is considered to be met for transition. **Tags:**atp.Status=draft
minDeadline	TimeValue	1	attr	Defines the shortest time span after which the deadline is considered to be met for transition. **Tags:**atp.Status=draft

Table 9.24: DeadlineSupervision

9.3.4 Global supervision entity deployment

The `GlobalSupervision` definition of supervision for the Platform Health Management is a second level supervision which takes the status of one or several `LocalSupervision`s (with their respective `AliveSupervision`s, `LogicalSupervision`s, and `DeadlineSupervision`s) and aggregates the individual states of these `LocalSupervision`s into one global supervision status (see also figure 9.10).

[TPS_MANI_03513]{DRAFT} **Collection of `LocalSupervision`s into a global supervision** ⌈All referenced `LocalSupervision`s in the scope of `GlobalSupervision.localSupervision` shall be taken into the aggregation of the status of the `GlobalSupervision`.⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03512]{DRAFT} **Applicability of global supervision without `Process` context** ⌈The referenced `LocalSupervision`s contributing to a specific `GlobalSupervision` may refer to `SupervisionCheckpoint`s where each `SupervisionCheckpoint` may refer to a different `Process`.⌋*(RS_MANI_00023, RS_MANI_00032)*

With [TPS_MANI_03512] the `GlobalSupervision` and the reference `GlobalSupervision.localSupervision` can be used to establish two use-cases:

- compose the status of one `Executable` instance (`Process`) in case all referenced `LocalSupervision`s are defined with the same `Process` context
- compose the status of several `Executable` instances (`Process`es) in case the referenced `LocalSupervision`s are defined with (partially) different `Process` contexts.

[constr_3537]{DRAFT} **LocalSupervision referenced once in the context of a GlobalSupervision** ⌈Any LocalSupervision shall be referenced at most once by a GlobalSupervision in the role GlobalSupervision.localSupervision.⌋ *()*

[constr_3537] associates a LocalSupervision to up to one GlobalSupervision.

[TPS_MANI_03514]{DRAFT} **Expiration tolerance for GlobalSupervision** ⌈The attribute GlobalSupervision.expiredSupervisionCyclesTolerance defines how many supervision cycles this incorrect global supervision is maintained in the state *expired* before it is considered *stopped*.⌋ *(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03552]{DRAFT} **Supervision cycle for GlobalSupervision** ⌈The attribute GlobalSupervision.supervisionCycle defines at which rate the GlobalSupervision shall be monitored.⌋ *(RS_MANI_00023, RS_MANI_00032)*

[constr_1738]{DRAFT} **Multiplicity of reference GlobalSupervision.localSupervision** ⌈At the time of deployment of a GlobalSupervision, at least one reference to meta-class LocalSupervision in the role localSupervision shall exist.⌋ *()*

Class	**GlobalSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a collection of LocalSupervisions in order to provide a aggregated supervision state. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
expired Supervision Cycles Tolerance	PositiveInteger	0..1	attr	Defines the acceptable amount of cycles with EXPIRED supervision status of this GlobalSupervision before it is considered STOPPED. **Tags:**atp.Status=draft
local Supervision	LocalSupervision	*	ref	Reference to the LocalSupervisions which are used to derive the status of this GlobalSupervision. **Tags:**atp.Status=draft
supervision Cycle	TimeValue	1	attr	Defines at which cycle the GlobalSupervision shall be executed.

Table 9.25: GlobalSupervision

[constr_1809]{DRAFT} **Global supervision restricted to one function group** ⌈Within the context of one GlobalSupervision, all LocalSupervisions referenced in the role localSupervision shall only aggregate aliveSupervision, logicalSupervision, and deadlineSupervision where all targets of the references

- aliveSupervision.checkpoint.process
- logicalSupervision.initialCheckpoint.process
- logicalSupervision.finalCheckpoint.process
- logicalSupervision.transition.source.process

- logicalSupervision.transition.target.process
- deadlineSupervision.checkpointTransition.source.process
- deadlineSupervision.checkpointTransition.target.process

aggregates a stateDependentStartupConfig where the reference in the role functionGroupState.contextModeDeclarationGroupPrototype refers to the exact same ModeDeclarationGroupPrototype (that implements the function group, as far as state management is concerned)⌋()

The model relations relevant for [constr_1809] are sketched in Figure 9.11:

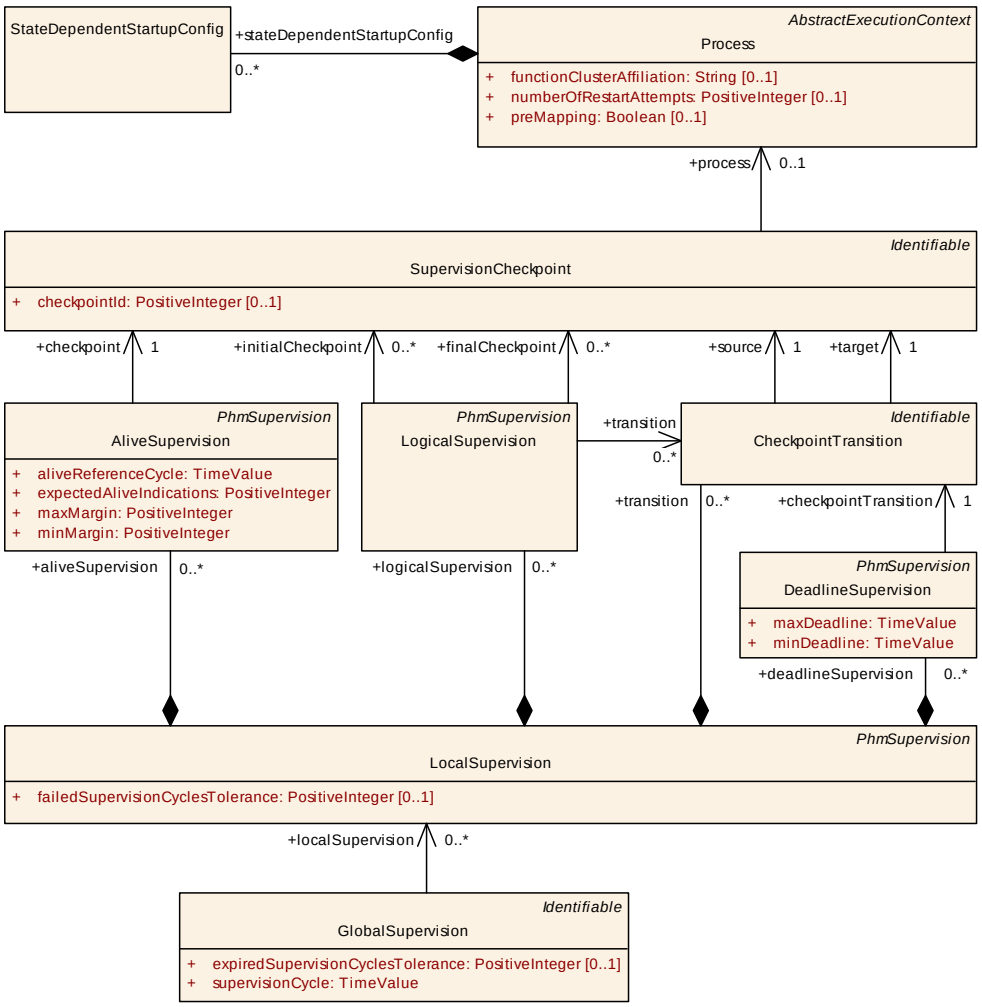


Figure 9.11: Visualization for the restriction of global supervision to one function group

9.3.5 Health channel deployment

The HealthChannel is used as an abstraction to the Platform Health Management input for the RecoveryNotification.

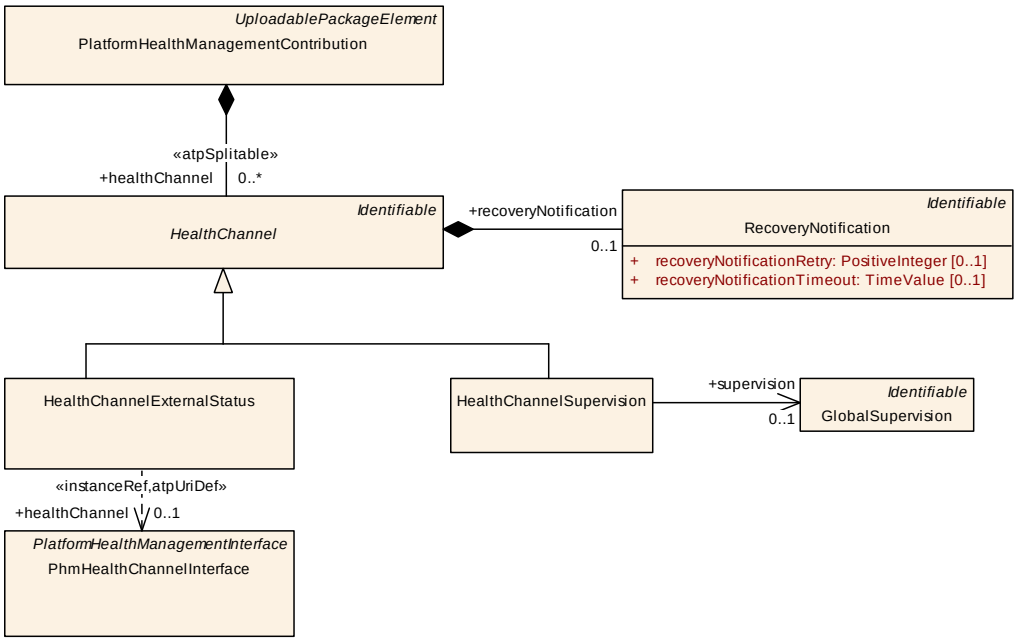


Figure 9.12: Modeling of `HealthChannel`

Class	***HealthChannel*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines the source of a health channel. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	HealthChannelExternalStatus, HealthChannelSupervision			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
recovery Notification	RecoveryNotification	0..1	aggr	Defines the RecoveryNotification. **Tags:**atp.Status=draft

Table 9.26: HealthChannel

The specialized use-cases for `HealthChannel`s are described in the following sections. A `HealthChannel` can either represent

- the status of a `GlobalSupervision` using the `HealthChannelSupervision` or
- the status of a reported `PhmHealthChannelInterface` using the `HealthChannelExternalStatus`

9.3.5.1 Supervision health channel deployment

The `HealthChannelSupervision` is used to take the status of a `GlobalSupervision` trigger the `RecoveryNotification` in case a violation is detected by the Phm.

[TPS_MANI_03516]{DRAFT} **Status for `HealthChannelSupervision`** ⌈The status of the `GlobalSupervision` which is referenced in the role `supervision` is taken as the trigger for the `RecoveryNotification`.⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**HealthChannelSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a health channel representing the status of a PhmSupervision. **Tags:**atp.Status=draft			
Base	*ARObject*, *HealthChannel*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
supervision	GlobalSupervision	0..1	ref	Reference to the GlobalSupervision as source for the health channel. **Tags:**atp.Status=draft

Table 9.27: HealthChannelSupervision

9.3.5.2 External health channel deployment

The `HealthChannelExternalStatus` is used to define a list of `HealthChannelExternalReportedStatus` in the role `healthChannel`. If for this `HealthChannelExternalStatus` and the referenced `Process` a status is reported which is part of the list in `healthChannel` then this `HealthChannelExternalStatus` is considered violated and a notification to the respective `RecoveryNotification` is performed.

[TPS_MANI_03545]{DRAFT} **Existence of `HealthChannelExternalStatus`** ⌈For each `RPortPrototype` typed by a `PhmHealthChannelInterface` there may be a `HealthChannelExternalStatus` defined.⌋*(RS_MANI_00023, RS_MANI_00032)*

The attribute `HealthChannelExternalReportedStatus.statusId` replicates the value of the referenced `PhmHealthChannelStatus.statusId`. During deployment the `PhmHealthChannelInterface` and its content is no longer available and therefore needs to be made available to the Phm.

[TPS_MANI_03625]{DRAFT} **Consistency of `HealthChannelExternalReportedStatus.statusId` and `PhmHealthChannelStatus.statusId`** ⌈The value of `HealthChannelExternalReportedStatus.statusId` shall be identical to the value of `PhmHealthChannelStatus.statusId` which is referenced in `HealthChannelExternalReportedStatus.status`.⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03546]{DRAFT} **Definition of reported health status `RPortPrototype`** ⌈The `RPortPrototype` typed by a `PhmHealthChannelInterface` is used to report the status of a health channel by the application software. This specific `RPortPrototype` is defined as the `contextRPortPrototype` of the instance

reference HealthChannelExternalStatus.healthChannel.⌋(*RS_MANI_00023, RS_MANI_00032*)

[TPS_MANI_03517]{DRAFT} **Evaluation of HealthChannelExternalStatus** ⌈The reported value of the HealthChannelExternalStatus according to [TPS_MANI_03546] will be compared to the list of statuses provided in notifiedStatus. If the reported status value matches one of the listed statusIds then this HealthChannelExternalStatus is considered violated and the respective RecoveryNotification is issued.⌋(*RS_MANI_00023, RS_MANI_00032*)

[TPS_MANI_03553]{DRAFT} **Applicability of health channel to a specific Process** ⌈The reference HealthChannelExternalStatus.process defines to which specific Process this HealthChannelExternalStatus definition shall be applied to.⌋(*RS_MANI_00023, RS_MANI_00032*)

This means that only if a PhmHealthChannelStatus is reported from the context of this Process it is considered to be this HealthChannelExternalStatus.

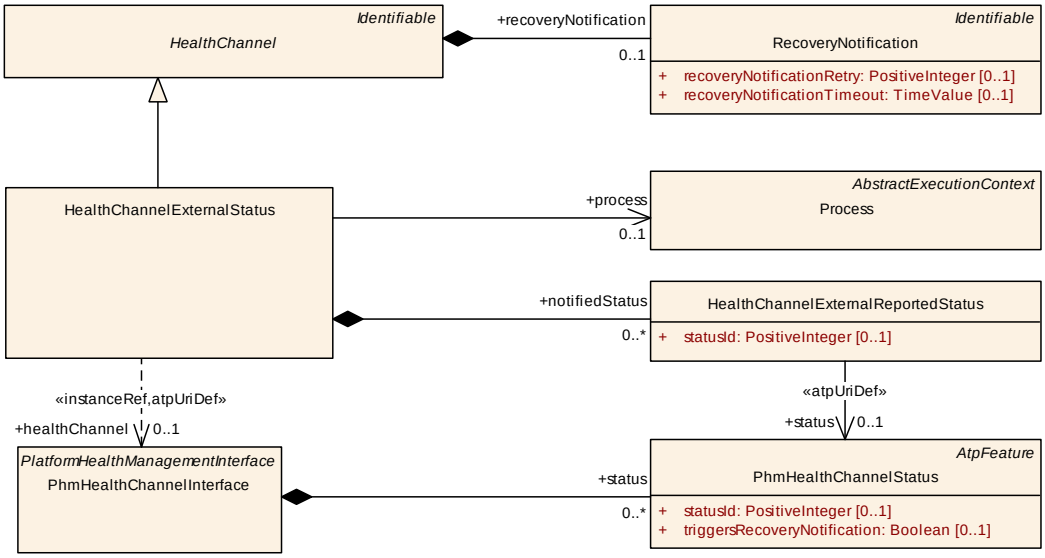


Figure 9.13: Modeling of HealthChannelExternalStatus

Class	**HealthChannelExternalStatus**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a health channel representing the status of an external health channel. **Tags:**atp.Status=draft			
Base	*ARObject*, *HealthChannel*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
healthChannel	PhmHealthChannelInterface	0..1	iref	Refers to the HealthChannel. **Tags:**atp.Status=draft **InstanceRef implemented by:**PhmHealthChannelInExecutableInstanceRef

△

Class	**HealthChannelExternalStatus**			
notifiedStatus	HealthChannelExternal ReportedStatus	*	aggr	This is a list of statuses which shall trigger the Recovery Notification of this HealthChannelExternalStatus. **Tags:**atp.Status=draft
process	Process	0..1	ref	Defines the Process this Health Channel shall be monitored. **Tags:**atp.Status=draft

Table 9.28: HealthChannelExternalStatus

Class	**HealthChannelExternalReportedStatus**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a health channel representing the status of an external health channel. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
status	PhmHealthChannel Status	0..1	ref	Reference to one status of a PhmHealthChannel. **Stereotypes:** atpUriDef **Tags:**atp.Status=draft
statusId	PositiveInteger	0..1	attr	Defines the numeric value which is used to identify the reporting of this HealthChannelExternalReportedStatus to the Phm. **Tags:**atp.Status=draft

Table 9.29: HealthChannelExternalReportedStatus

9.3.6 Recovery Notification

If Phm detects a Supervision violation or Health Channel violation then the associated `RecoveryNotification` at the `HealthChannel` is activated. This general setup is illustrated in figure 3.42. Via the `RecoveryNotificationToPPortPrototypeMapping` this `RecoveryNotification` is mapped to a dedicated `PPortPrototype` in the context of a dedicated `Process` implementing the State Management functionality.

[constr_3612]{DRAFT} **Multiplicity of references `recoveryNotification`, `recoveryAction`, and `process` at `RecoveryNotificationToPPortPrototypeMapping`** ⌈The references `recoveryNotification`, `recoveryAction`, and `process` shall be defined for each `RecoveryNotificationToPPortPrototypeMapping` **at the time when manifest creation is finished.**⌋*()*

[constr_3613]{DRAFT} **Reference to a `PhmSupervisionRecoveryNotificationInterface` in the context of a `HealthChannelSupervision`** ⌈If the `RecoveryNotification` is aggregated by a `HealthChannelSupervision` then the `RecoveryNotificationToPPortPrototypeMapping` shall refer to a `PPortPrototype` in the role `recoveryAction` typed by `PhmSupervisionRecoveryNotificationInterface`.⌋*()*

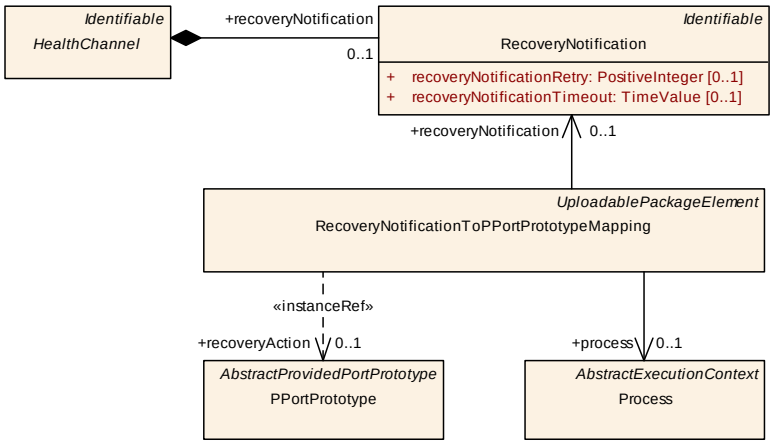


Figure 9.14: Modeling of `RecoveryNotification`

[constr_3614]{DRAFT} **Reference to a `PhmHealthChannelRecoveryNotificationInterface` in the context of a `HealthChannelExternalStatus`** ⌈If the `RecoveryNotification` is aggregated by a `HealthChannelExternalStatus` then the `RecoveryNotificationToPPortPrototypeMapping` shall refer to a `PPortPrototype` in the role `recoveryAction` typed by `PhmHealthChannelRecoveryNotificationInterface`.⌋*()*

Class	**RecoveryNotification**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This meta-class represents a PHM action that can trigger a recovery operation inside a piece of State Management software. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
recovery Notification Retry	PositiveInteger	0..1	attr	Number of times Platform Health management tries sends a notification to State Management again before triggering a watchdog reaction. **Tags:**atp.Status=draft
recovery Notification Timeout	TimeValue	0..1	attr	The maximum acceptable amount of time (in seconds), Platform Health Management waits for an acknowledgment by State Management after sending the notification. **Tags:**atp.Status=draft

Table 9.30: RecoveryNotification

Class	**RecoveryNotificationToPPortPrototypeMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement
Note	This meta-class represents the ability to associate a RecoveryNotification to a PPortPrototype while also being able to identify the respective Process in which the actual recovery executes. **Tags:** atp.Status=draft atp.recommendedPackage=RecoveryNotificationMappings

▽

△

Class	**RecoveryNotificationToPPortPrototypeMapping**			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
process	Process	0..1	ref	Reference to the process which represents the State Management instance that the revovery notification shall be applied to. **Tags:**atp.Status=draft
recoveryAction	PPortPrototype	0..1	iref	This reference identifies the PortPrototype to be addressed as part of a PHM recovery. **Tags:**atp.Status=draft **InstanceRef implemented by:**PPortPrototypeIn ExecutableInstanceRef
recovery Notification	RecoveryNotification	0..1	ref	This reference identifies the applicable Recovery Notification to be mapped. **Tags:**atp.Status=draft

Table 9.31: RecoveryNotificationToPPortPrototypeMapping

9.4 Time Synchronization Deployment

9.4.1 Overview

This chapter explains the configuration of the Time Synchronization functional cluster.

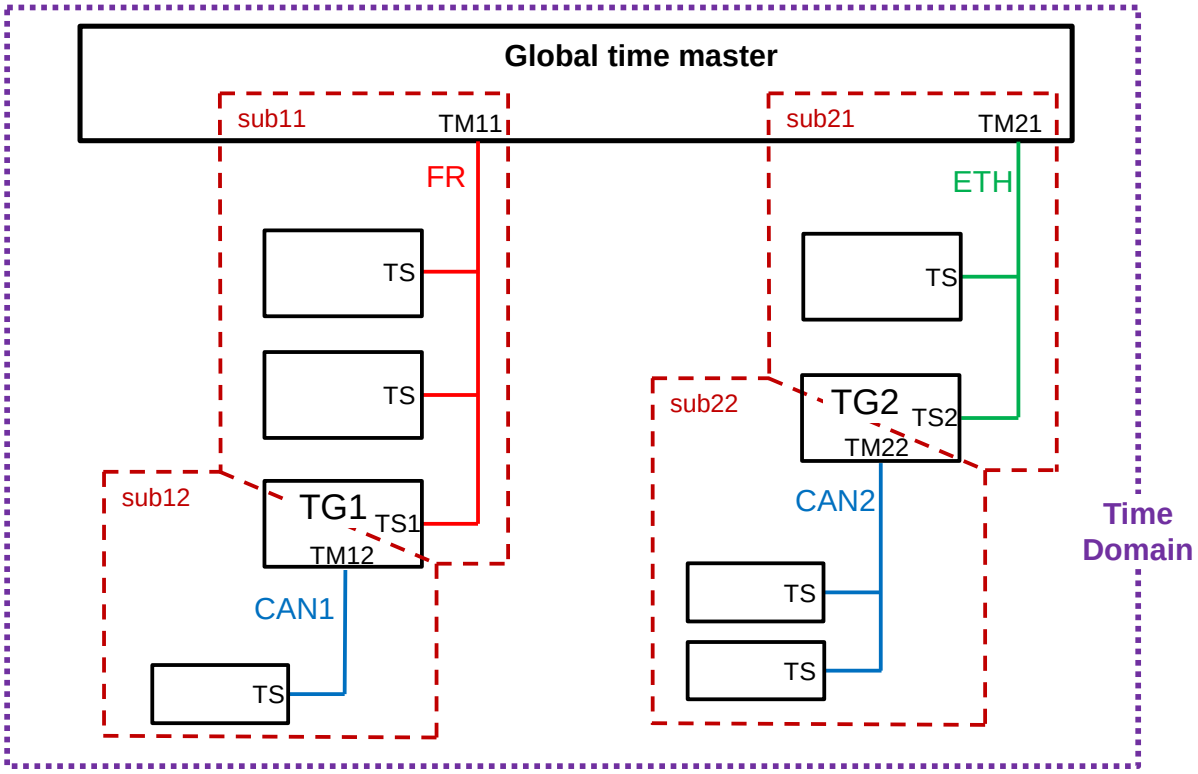
An adaptive AUTOSAR application can utilize several (synchronized) Time-Base Resources which are provided by the Time Synchronization functional cluster [11].

The intended interaction of an adaptive AUTOSAR application with Time Synchronization is described in chapter 3.9.

Since an adaptive `Machine` is usually collaborating with other `Machine`s (adaptive) and ECUs (classic), special focus has been put on the vehicle wide definition of synchronized time.

For a detailed specification please refer to the *Global Time Synchronization* chapter in the *System Template* [17].

Figure 9.15 provides an example system view on time domains and their transportation over diverse networks. In the scope of the *AUTOSAR adaptive platform* the focus is put on the Ethernet interaction with the rest of the system.



TS: GlobalTime slave
TM: GlobalTime master
TG: GlobalTime gateway

Figure 9.15: Example setup of Synchronized Global Time in AUTOSAR

9.4.2 Time Synchronization functional cluster configuration

The representation of the Time Synchronization functional cluster [11] within one specific `Machine` is defined by the `TimeSyncModuleInstantiation`. The `Machine` has the ability to define a set of `moduleInstantiations`, where a specialization can be the `TimeSyncModuleInstantiation`.

Class	**TimeSyncModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class defines the attributes for the Time Synchronization configuration on a specific machine. **Tags:**atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
timeBase	TimeBaseResource	*	aggr	This aggregation defines the configured Time Bases for Time Synchronization. **Tags:**atp.Status=draft

Table 9.32: TimeSyncModuleInstantiation

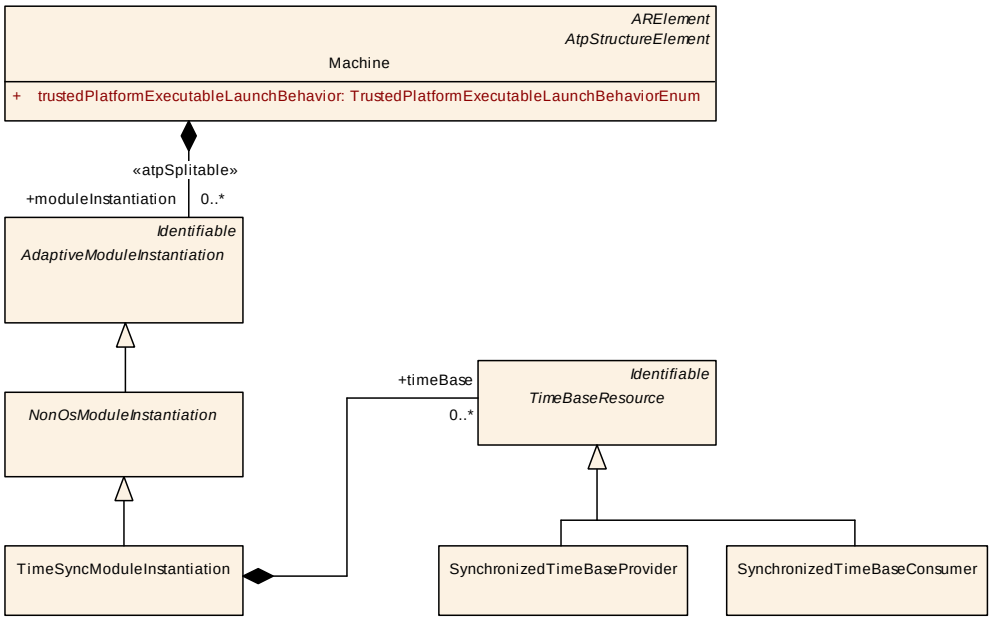


Figure 9.16: Modeling of `TimeSyncModuleInstantiation`

9.4.3 Time Base

The `TimeSyncModuleInstantiation` represents the actual instance of the Time Synchronization functional cluster executed on a specific `Machine`. In the scope of the `TimeSyncModuleInstantiation` the Time-Base Resources are defined.

[TPS_MANI_03539]{DRAFT} **Definition of Time-Base Resources** ⌈The meta-class `TimeSyncModuleInstantiation` has the ability to define a set of Time-Base Resources of kind `TimeBaseResource` in the role `timeBase`.⌋*(RS_MANI_00040)*

There are several sub types of `TimeBaseResource` which will be explained in the following sections.

Class	***TimeBaseResource*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents the attributes of one Time Base Resource for Time Synchronization. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	SynchronizedTimeBaseConsumer, SynchronizedTimeBaseProvider			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 9.33: TimeBaseResource

9.4.3.1 Synchronized time base

When configuring a synchronized time base many configuration aspects are already provided by the definition of the GlobalTimeDomain and are specified in the *System Template* [17] and associated to MachineDesign.

As for the configuration of the TimeSyncModuleInstantiation the usage of the SynchronizedTimeBaseProvider respectively SynchronizedTimeBaseConsumer defines the interaction with the GlobalTimeDomain.

[TPS_MANI_03541]{DRAFT} **Definition of SynchronizedTimeBaseConsumer** ⌈The meta-class SynchronizedTimeBaseConsumer defines a Time Base which is synchronized with a time coming from the network. With the reference SynchronizedTimeBaseConsumer.networkTimeConsumer to a GlobalTimeSlave the relation to the system model is established.⌋*(RS_MANI_00040)*

[TPS_MANI_03542]{DRAFT} **Definition of SynchronizedTimeBaseProvider** ⌈The meta-class SynchronizedTimeBaseProvider defines a Time Base which is propagated to a time on the network. With the reference SynchronizedTimeBaseProvider.networkTimeProvider to a GlobalTimeMaster the relation to the system model is established.⌋*(RS_MANI_00040)*

Some aspects of the Synchronized Time Base for the *provider* role are not available in the system model, those are provided with the TimeSyncCorrection.

[TPS_MANI_03543]{DRAFT} **Definition of time sync correction attributes** ⌈The meta-class TimeSyncCorrection defines the attributes required to specify the time sync correction behavior of a SynchronizedTimeBaseProvider. The SynchronizedTimeBaseProvider aggregates the TimeSyncCorrection in the role timeSyncCorrection.⌋*(RS_MANI_00040)*

The synchronized global time feature also supports the definition of *offset* time domains.

[TPS_MANI_03547]{DRAFT} **Definition of *offset* time domains** ⌈A GlobalTimeDomain which has a offsetTimeDomain reference defined is considered an *offset* time domain. The reference source is the *offset* time domain. The reference *target* is the synchronized time domain.⌋*(RS_MANI_00040)*

The *offset* time domain is applicable to GlobalTimeMaster (therefore also SynchronizedTimeBaseProvider) and GlobalTimeSlave (therefore also SynchronizedTimeBaseConsumer).

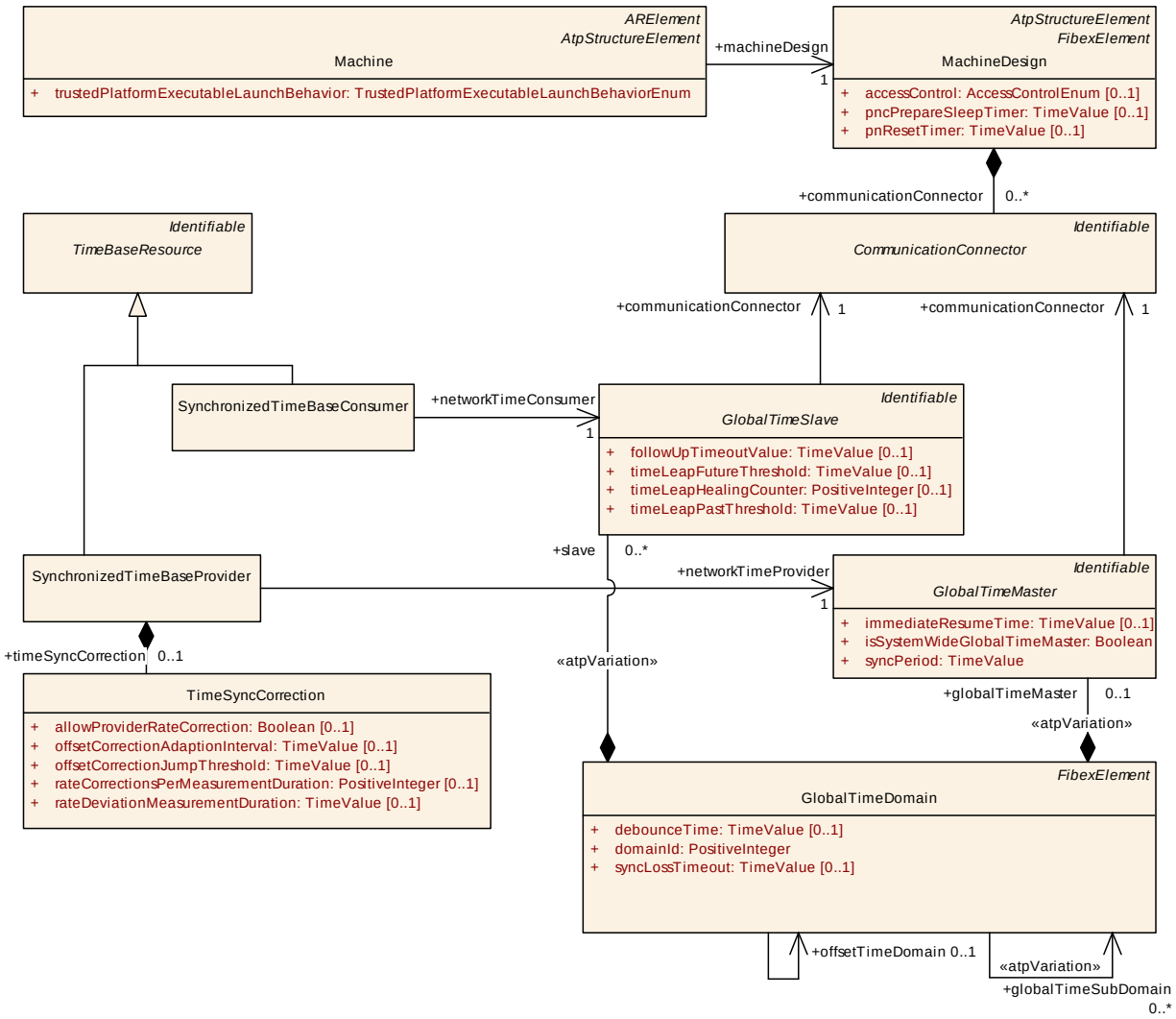


Figure 9.17: Modeling of synchronized time bases

Class	**SynchronizedTimeBaseConsumer**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents a Synchronized Time Base Consumer. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimeBaseResource*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
networkTime Consumer	GlobalTimeSlave	1	ref	This reference defines the GlobalTime Consumer which is synchronized with this Time Base. **Tags:**atp.Status=draft

Table 9.34: SynchronizedTimeBaseConsumer

Class	SynchronizedTimeBaseProvider			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents a Synchronized Time Base Provider. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimeBaseResource*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
networkTime Provider	GlobalTimeMaster	1	ref	This reference defines the GlobalTime Provider which is synchronized with this Time Base. **Tags:**atp.Status=draft
timeSync Correction	TimeSyncCorrection	0..1	aggr	This aggregation defines the attributes used for the correction of time synchronization. **Tags:**atp.Status=draft

Table 9.35: SynchronizedTimeBaseProvider

Class	TimeSyncCorrection			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents the attributes used for the correction of time synchronization. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
allowProvider RateCorrection	Boolean	0..1	attr	Defines whether the rate correction value of a Time Base can be set by means of the method setRateCorrection(). false: rate correction cannot be set by method setRate Correction(). true: rate correction can be set by method setRate Correction(). **Tags:**atp.Status=draft
offsetCorrection AdaptionInterval	TimeValue	0..1	attr	Defines the interval during which the adaptive rate correction cancels out the rate and time deviation. Unit: seconds. **Tags:**atp.Status=draft
offsetCorrection JumpThreshold	TimeValue	0..1	attr	Threshold for the correction method. Deviations below this value will be corrected by a linear reduction over a defined timespan. Values equal and greater than this value will be corrected by immediately setting the correct time and rate in form of a jump. Unit: seconds. **Tags:**atp.Status=draft
rateCorrections Per Measurement Duration	PositiveInteger	0..1	attr	Number of simultaneous rate measurements to determine the current rate deviation. **Tags:**atp.Status=draft
rateDeviation Measurement Duration	TimeValue	0..1	attr	Time span used to calculate the rate deviation. Unit: seconds. **Tags:**atp.Status=draft

Table 9.36: TimeSyncCorrection

9.4.3.2 Persistent Time Base value storage

The Time Synchronization functional cluster allows to define a storage for a SynchronizedTimeBaseProvider time base. During shutdown the value of that specific time base is stored in a PersistencyDeploymentElement (either PersistencyKeyValuePair or PersistencyFile) and during startup the value is read from persistency and restored at the time base.

[TPS_MANI_03632]{DRAFT} **Semantics of TimeBaseProviderToPersistencyMapping** ⌈The TimeBaseProviderToPersistencyMapping defines that the referenced SynchronizedTimeBaseProvider (TimeBaseProviderToPersistencyMapping.timeBaseProvider) time value shall be stored to persistency during shutdown and restored from persistency during startup (TimeBaseProviderToPersistencyMapping.persistencyDeploymentElement).⌋*(RS_MANI_00040)*

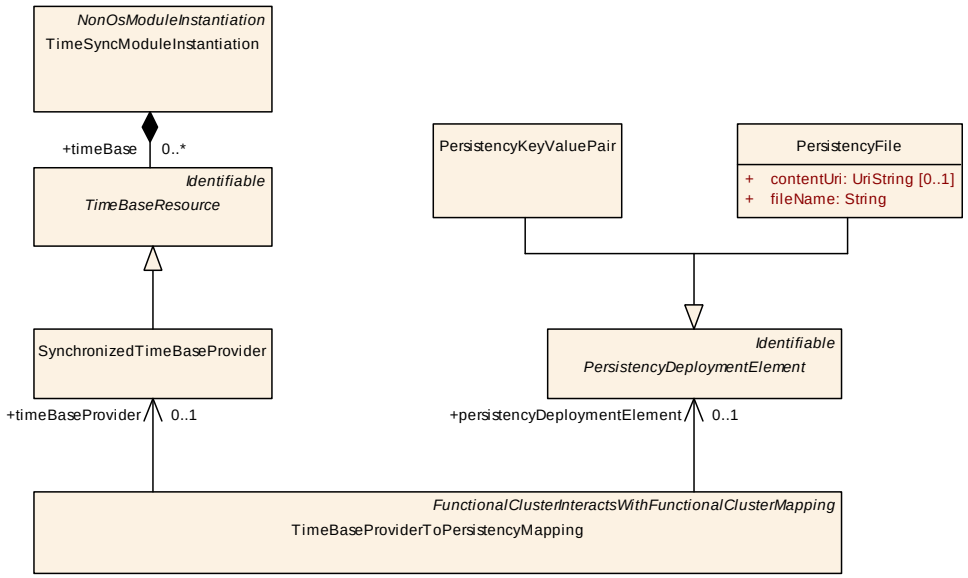


Figure 9.18: Modeling of TimeBaseProviderToPersistencyMapping

Class	**TimeBaseProviderToPersistencyMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents the ability to define a mapping between a TimeBaseProvider and a PersistencyDeploymentElement for the purpose of storing and retrieving the time value. **Tags:** atp.Status=draft atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
persistency Deployment Element	PersistencyDeployment Element	0..1	ref	This reference represents the PersistencyDeployment Element where the time value shall be stored in and retrieved from. **Tags:**atp.Status=draft

△

Class	**TimeBaseProviderToPersistencyMapping**			
timeBase Provider	SynchronizedTimeBase Provider	0..1	ref	This reference represents the mapped TimeBase Provider. **Tags:**atp.Status=draft

Table 9.37: TimeBaseProviderToPersistencyMapping

[constr_3619]{DRAFT} **Mandatory references of `TimeBaseProviderToPersistencyMapping`** ⌈The references `TimeBaseProviderToPersistencyMapping.persistencyDeploymentElement` and `TimeBaseProviderToPersistencyMapping.timeBaseProvider` shall exist **at the time when manifest creation is finished.**⌋*()*

9.4.3.3 Ethernet synchronized time

As the *AUTOSAR adaptive platform* supports Ethernet as communication network also the time synchronization using Ethernet is supported.

In order to configure the behavior of the Ethernet time synchronization the specific sub-classes are used as shown in figure 9.19.

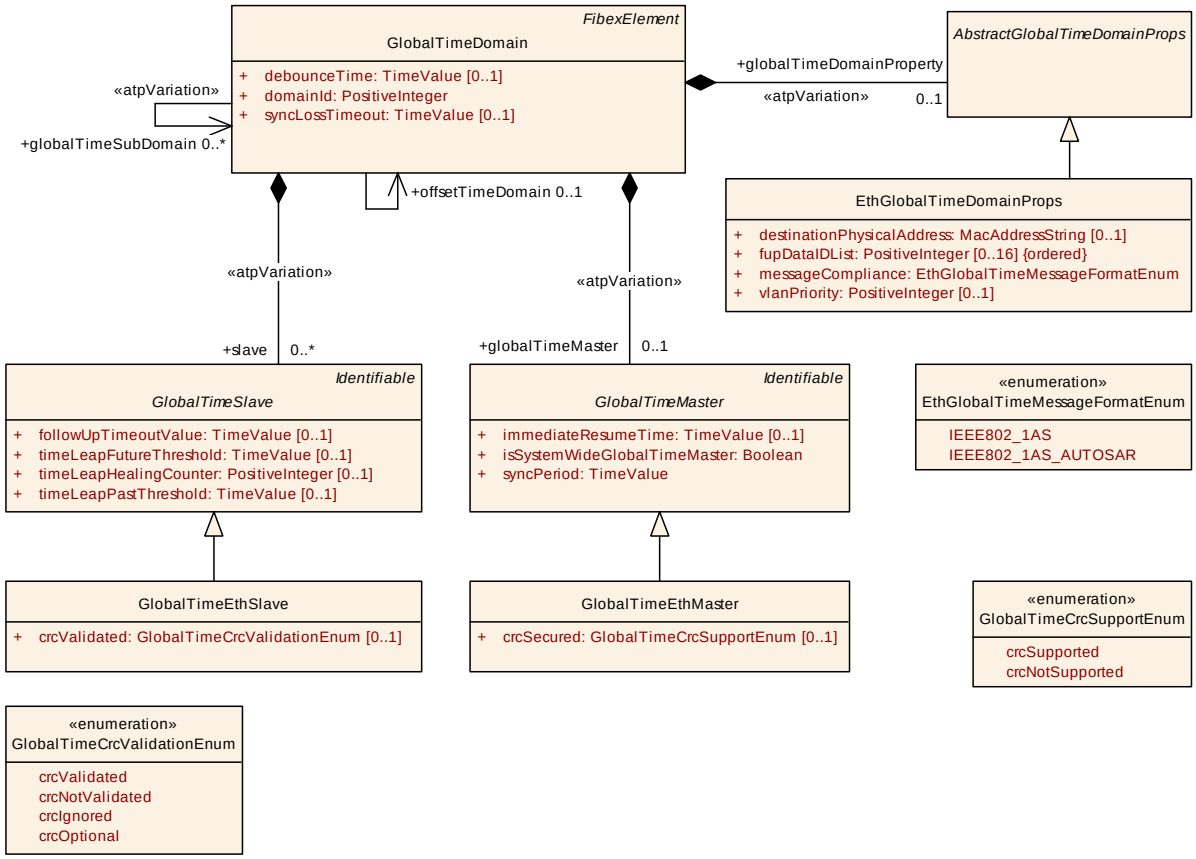


Figure 9.19: Modeling of Ethernet synchronized time

Class	**EthGlobalTimeDomainProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime::ETH			
Note	Enables the definition of Ethernet Global Time specific properties.			
Base	*ARObject*, *AbstractGlobalTimeDomainProps*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
crcFlags	EthTSynCrcFlags	0..1	aggr	Defines the fields of the message which shall be taken into account for CRC calculation and verification.
destination Physical Address	MacAddressString	0..1	attr	Defines the MAC multicast address the Ethernet time sync messages are communicated on.
fupDataIDList (ordered)	PositiveInteger	0..16	attr	The DataIDList for FUP messages to calculate CRC.
managed CouplingPort	EthGlobalTime ManagedCouplingPort	*	aggr	Collection of CouplingPorts which are managed in the scope of this Ethernet GlobalTimeDomain.
message Compliance	EthGlobalTimeMessage FormatEnum	1	attr	Defines the compliance of the Ethernet time sync messages to specific standards.
vlanPriority	PositiveInteger	0..1	attr	Defines which VLAN priority shall be assigned to a time sync message in case the message is sent using a VLAN tag.

Table 9.38: EthGlobalTimeDomainProps

Class	**GlobalTimeEthSlave**			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime::ETH			
Note	This represents the specialization of the GlobalTimeSlave for Ethernet communication.			
Base	*ARObject*, *GlobalTimeSlave*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
crcValidated	GlobalTimeCrc ValidationEnum	0..1	attr	Definition of whether or not validation of the CRC is supported.

Table 9.39: GlobalTimeEthSlave

Class	**GlobalTimeEthMaster**			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime::ETH			
Note	This represents the specialization of the GlobalTimeMaster for Ethernet communication.			
Base	*ARObject*, *GlobalTimeMaster*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
crcSecured	GlobalTimeCrcSupport Enum	0..1	attr	Definition of whether or not CRC is supported. This is only relevant for selected bus systems.
subTlvConfig	EthTSynSubTlvConfig	0..1	aggr	Defines the subTLV fields which shall be included in the time sync message.

Table 9.40: GlobalTimeEthMaster

9.4.4 Time Base to Port Prototype mapping

The `TimeBaseResource` definition of chapter 9.4.3 and the `RPortPrototype` typed by a sub-class of `AbstractSynchronizedTimeBaseInterface` of chapter 3.9 have to be mapped to each other in order to define the binding of application software to the platform foundation software implementing the time synchronization.

[TPS_MANI_03548]{DRAFT} **Definition of `TimeSyncPortPrototypeToTimeBaseMapping`** ⌈A `TimeSyncPortPrototypeToTimeBaseMapping` is used to define a mapping between a `TimeBaseResource` and a `RPortPrototype` typed by a sub-class of `AbstractSynchronizedTimeBaseInterface` in the context of a `Process`.⌋*(RS_MANI_00040)*

The `TimeSyncPortPrototypeToTimeBaseMapping` takes the `Process` into account so that every instantiation of an `Executable` (and the resulting instantiation of all the `RPortPrototype`s typed by a sub-class of `AbstractSynchronizedTimeBaseInterface`) can be mapped individually to `TimeBaseResource`s.

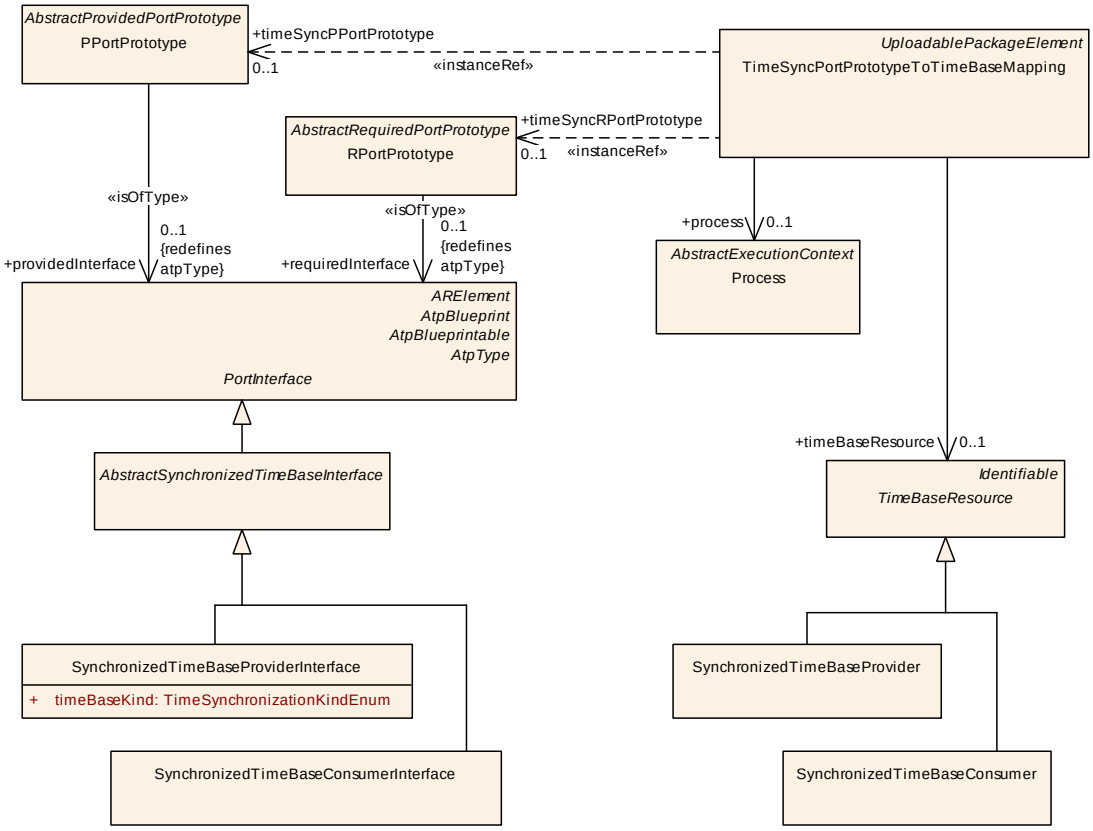


Figure 9.20: Modeling of `TimeSyncPortPrototypeToTimeBaseMapping`

Class	**TimeSyncPortPrototypeToTimeBaseMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class provides the ability to map a PortPrototype typed by a AbstractSynchronizedTimeBase Interface to a TimeBaseResource in the context of a Process. **Tags:** atp.Status=draft atp.recommendedPackage=TimeSyncPortPrototypeToTimeBaseMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	TimeSyncPortPrototypeToTimeBaseMapping			
process	Process	0..1	ref	Reference to the context Process this mapping applies to. **Tags:**atp.Status=draft
timeBase Resource	TimeBaseResource	0..1	ref	Reference to the mapped TimeBaseResource. **Tags:**atp.Status=draft
timeSyncPPort Prototype	PPortPrototype	0..1	iref	Instance reference to the mapped PPortPrototype typed by a AbstractSynchronizedTimeBaseInterface. **Tags:**atp.Status=draft **InstanceRef implemented by:**PPortPrototypeIn ExecutableInstanceRef
timeSyncRPort Prototype	RPortPrototype	0..1	iref	Instance reference to the mapped RPortPrototype typed by a AbstractSynchronizedTimeBaseInterface. **Tags:**atp.Status=draft **InstanceRef implemented by:**RPortPrototypeIn ExecutableInstanceRef

Table 9.41: TimeSyncPortPrototypeToTimeBaseMapping

The example shown in figure 9.21 illustrates the mapping of `RPortPrototype`s typed by one of the sub-classes of `AbstractSynchronizedTimeBaseInterface` to actually configured `TimeBaseResource`s at the Time Sync Management.

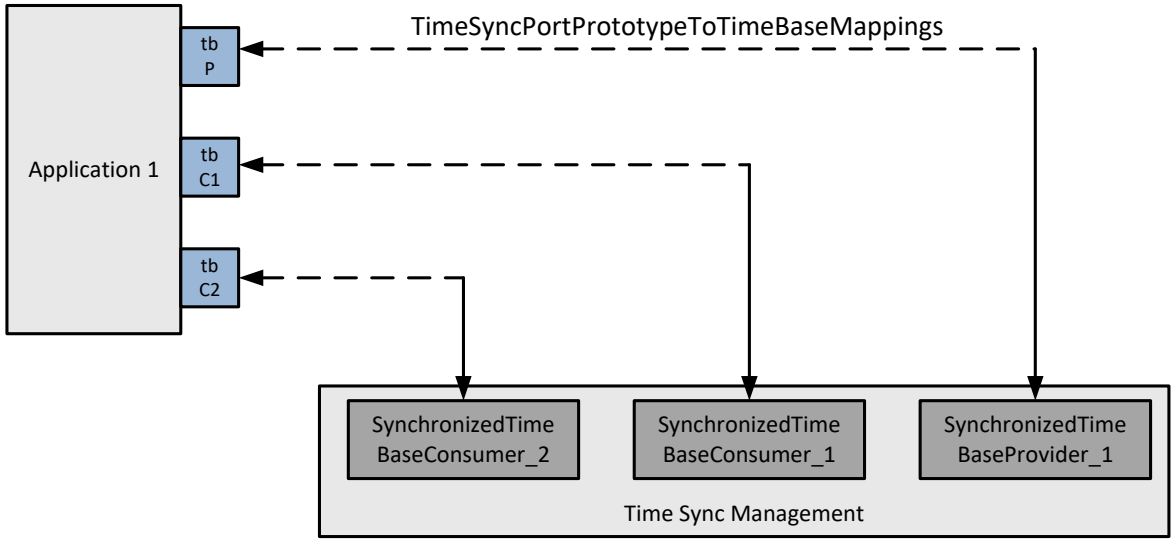


Figure 9.21: Example PortPrototype to TimeBase mapping

9.5 DoIP configuration

[TPS_MANI_03164]{DRAFT} **`Machine`-specific configuration settings for DoIP** ⌈The `Machine`-specific configuration settings for DoIP are collected in `DoIpInstantiation`.⌋*(RS_MANI_00023)*

Class	**DoIpInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class defines the attributes for the DoIP configuration on a specific machine. **Tags:**atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *NonOsModule Instantiation*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eid	PositiveUnlimitedInteger	0..1	attr	Configured EID (Entity ID) used for VehicleIdentification Request. If configured, take this value, if not configured use MAC address.
entityStatusMax ByteFieldUse	Boolean	1	attr	This attribute is used to distinguish the optional support of the Max data size element of a diagnostic entity status response.
gid	PositiveUnlimitedInteger	0..1	attr	Configured GID (Group ID) used for VehicleIdentification Request. If configured, take this value (and set "Further action required" byte to 0x00="No further action required"), if not configured use ServiceInterface Do IPGroupIdentification to retrieve GID and 'further action required' values.
gidInvalidity Pattern	PositiveInteger	1	attr	Specifies the Byte pattern that is used for response messages if no valid GID could be retrieved. Only the value '0' or '255' is allowed.
logicalAddress	PositiveInteger	1	attr	Describes the logical address of the DoIP entity, which is used for VehicleAnnouncement and RoutingActivation responses.
maxRequest Bytes	PositiveInteger	1	attr	Specifies the maximum allowed bytes of a DoIP message request without the DoIP header.
network Interface	DoIpNetwork Configuration	*	aggr	Network interface specific DoIP properties. **Tags:**atp.Status=draft
request Configuration	DoIpRequest Configuration	*	aggr	Request configuration that is used to determine whether an incoming DiagnosticMessage request needs to be interpreted as PHYSICAL or FUNCTIONAL. Any request with target address not within the configured target address range will be rejected. **Tags:**atp.Status=draft
vinInvalidity Pattern	PositiveInteger	1	attr	Specifies the Byte pattern that is used for response messages if no valid VIN could be retrieved. Only the value '0' or '255' is allowed.

Table 9.42: DoIpInstantiation

[constr_3425]{DRAFT} **Restriction of `DoIpInstantiation`s on a `Machine`**
⌈Each `Machine` shall aggregate at most one `DoIpInstantiation` in the role `moduleInstantiation`.⌋*()*

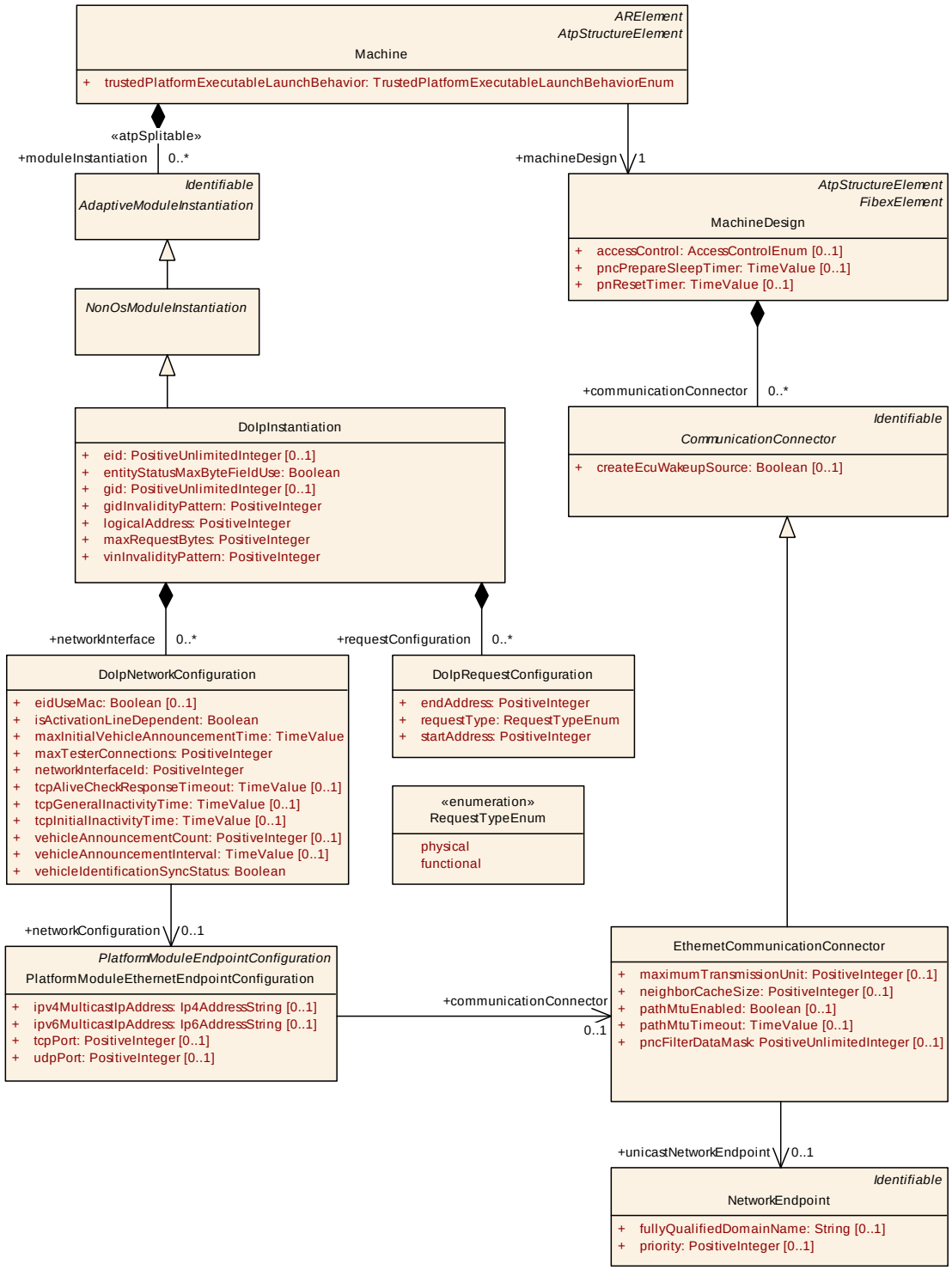


Figure 9.22: DoIP configuration

[constr_3495]{DRAFT} **Supported value range for attribute `DoIpInstantiation.eid`** ⌈The supported value range of attribute `DoIpInstantiation.eid` is limited to the interval [0..281474976710655].⌋*()*

[constr_3496]{DRAFT} **Supported value range for attribute DoIpInstantiation.gid** ⌈The supported value range of attribute DoIpInstantiation.gid is limited to the interval [0..281474976710655].⌋*()*

[constr_3497]{DRAFT} **Supported value range for attribute DoIpInstantiation.maxRequestBytes** ⌈The supported value range of attribute DoIpInstantiation.maxRequestBytes is limited to the interval [0..4294967295].⌋*()*

[constr_3498]{DRAFT} **Supported value range for attribute DoIpInstantiation.logicalAddress** ⌈The supported value range of attribute DoIpInstantiation.logicalAddress is limited to the interval [0..65535].⌋*()*

[TPS_MANI_03165]{DRAFT} **Network Interface configuration for DoIP** ⌈The DoIpNetworkConfiguration contains all configuration settings that are specific for a configured network connection. The network connection is configured with the PlatformModuleEthernetEndpointConfiguration that is referenced by the DoIpNetworkConfiguration in the role networkConfiguration.

The attributes tcpPort and udpPort are used to configure the Transport Protocol (Udp or Tcp) and the used Port number. The IP Address is configured in the NetworkEndpoint that is referenced by the PlatformModuleEthernetEndpointConfiguration via the EthernetCommunicationConnector.⌋*(RS_MANI_00023)*

Class	**DoIpNetworkConfiguration**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModuleImplementation			
Note	This element collects DoIP properties that are network interface specific. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eidUseMac	Boolean	0..1	attr	This attribute defines whther the MAC of the network interface is used as eid. True: MAC is used False: eid needs to be configured manually by DoIpInstantiation.eid.
isActivationLine Dependent	Boolean	1	attr	This attribute defines whether the network interface • is started "on-demand" when an activation line is sensed or • is always available.
maxInitial Vehicle Announcement Time	TimeValue	1	attr	Upper bound for the time to wait in [s] for sending first vehicle anouncement message after IP address assignment. Represents parameter A_DoIP_Announce_Wait of ISO 13400-2:2012. The value of this timing shall be determined randomly in the closed interval [0..max InitialVehicleAnnouncementTime].
maxTester Connections	PositiveInteger	1	attr	Maximum amount of tester connections that shall be maintained at one time before alive check is performed.
network Configuration	PlatformModule EthernetEndpoint Configuration	0..1	ref	Network configuration (Protocol, Port, IP Address) for transmission of DoIP messages on a specific VLAN. **Tags:**atp.Status=draft
network InterfaceId	PositiveInteger	1	attr	This attribute defines the identifier for the DoIPInterface.

▽

△

Class	DoIpNetworkConfiguration			
tcpAliveCheck Response Timeout	TimeValue	0..1	attr	Timeout in [s] for waiting for a response to an Alive Check request before the connection is considered to be disconnected. Represents parameter T_TCP_AliveCheck of ISO 13400-2:2012.
tcpGeneral InactivityTime	TimeValue	0..1	attr	Timeout in [s] for maximum inactivity of a TCP socket connection before the DoIP module will close the according socket connection. Represents parameter T_ TCP_General_Inactivity of ISO 13400-2:2012.
tcpInitial InactivityTime	TimeValue	0..1	attr	Timeout in [s] used for initial inactivity of a connected TCP socket connection directly after socket connection. Represents parameter T_TCP_Initial_Inactivity of ISO 13400-2:2012.
vehicle Announcement Count	PositiveInteger	0..1	attr	Number of vehicle announcement messages on IP address assignment. Represents parameter A_DoIP_ Announce_Num of ISO 13400-2:2012.
vehicle Announcement Interval	TimeValue	0..1	attr	Time to wait in [s] for sending subsequent vehicle anouncement messages. Represents parameter A_Do IP_Announce_Interval of ISO 13400-2:2012.
vehicle Identification SyncStatus	Boolean	1	attr	Defines if the optional VIN/GID synchronization status is used additionally in the vehicle identification/ announcement.

Table 9.43: DoIpNetworkConfiguration

Please note that it is possible to define several networkInterfaces in a DoIpInstantiation. For each network connection individual configuration settings can be set with the attributes that are defined in the DoIpNetworkConfiguration element, e.g. it is possible to configure the vehicle announcement for different network connections differently.

[constr_5046]{DRAFT} **Usage of DoIpNetworkConfiguration.eidUseMac** ⌈If DoIpInstantiation.eid is not configured, the value of DoIpNetworkConfiguration.eidUseMac shall be set to true.⌋*()*

[TPS_MANI_03218]{DRAFT} **Default value for the attribute tcpInitialInactivityTime of meta-class DoIpNetworkConfiguration** ⌈If no value for the attribute DoIpNetworkConfiguration.tcpInitialInactivityTime is defined then the default value of 2 seconds shall be assumed.⌋*(RS_MANI_00023)*

[TPS_MANI_03219]{DRAFT} **Default value for the attribute tcpGeneralInactivityTime of meta-class DoIpNetworkConfiguration** ⌈If no value for the attribute DoIpNetworkConfiguration.tcpGeneralInactivityTime is defined then the default value of 300 seconds shall be assumed.⌋*(RS_MANI_00023)*

[TPS_MANI_03220]{DRAFT} **Default value for the attribute vehicleAnnouncementCount of meta-class DoIpNetworkConfiguration** ⌈If no value for the attribute DoIpNetworkConfiguration.vehicleAnnouncementCount is defined then the default value of 3 shall be assumed.⌋*(RS_MANI_00023)*

[TPS_MANI_03221]{DRAFT} **Default value for the attribute vehicleAnnouncementInterval of meta-class DoIpNetworkConfiguration** ⌈If no value for the attribute DoIpNetworkConfiguration.vehicleAnnouncementInterval is defined then the default value of 0,5 seconds shall be assumed.⌋*(RS_MANI_00023)*

[TPS_MANI_03222]{DRAFT} **Default value for the attribute `tcpAliveCheckResponseTimeout` of meta-class `DoIpNetworkConfiguration`** ⌈If no value for the attribute `DoIpNetworkConfiguration.tcpAliveCheckResponseTimeout` is defined then the default value of 0,5 seconds shall be assumed.⌋*(RS_MANI_00023)*

During vehicle discovery the DoIP module responds by informing the tester about its own address, configured as the `logicalAddress`. The tester will approach the ECU under this UDS target address, thus the ECU should have a `SoftwareCluster` that is configured to respond to this UDS target address.

The list of available target addresses may or may not be obtainable from the `SoftwareCluster` with the `logicalAddress`.

In some cases, this `SoftwareCluster` may have the ability to inform the tester which other existing physical and/or logical addresses are available.

Class	**DoIpRequestConfiguration**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModuleImplementation			
Note	This meta-class specifies a range of target addresses and its interpretation as either physical or functional request. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
endAddress	PositiveInteger	1	attr	End address for range of target-addresses (including this address).
requestType	RequestTypeEnum	1	attr	Determines the type of request.
startAddress	PositiveInteger	1	attr	Start address for range of target-addresses (including this address).

Table 9.44: DoIpRequestConfiguration

[constr_3492]{DRAFT} **`DoIpInstantiation.logicalAddress` shall be defined as member in the `DoIpRequestConfiguration`** ⌈The `DoIpInstantiation.logicalAddress` shall be a member of the intervals of available physical addresses configured for the `DoIpInstantiation` in the `requestConfiguration`.⌋*()*

On top of that, there is the expectation that the configured `diagnosticAddress`es of `SoftwareCluster`s deployed to the `Machine` fit to the intervals defined in the context of the `DoIpInstantiation` in the `requestConfiguration`.

Please note that the `DoIpRequestConfiguration` corresponds to Table 39 that is defined in ISO-13400-2 [22].

[constr_3499]{DRAFT} **Supported value range for attribute `DoIpRequestConfiguration.startAddress`** ⌈The supported value range of attribute `DoIpRequestConfiguration.startAddress` is limited to the interval [0..65535].⌋*()*

[constr_5000]{DRAFT} **Supported value range for attribute `DoIpRequestConfiguration.endAddress`** ⌈The supported value range of attribute `DoIpRequestConfiguration.endAddress` is limited to the interval [0..65535].⌋*()*

[constr_3414]{DRAFT} **Allowed usage of `PlatformModuleEthernetEndpointConfiguration` attributes** ⌈Table 9.45 shows `PlatformModuleEthernetEndpointConfiguration` attributes that are allowed to be used to configure the network communication in the different platform modules.⌋ *()*

	Element	
`PlatformModuleEthernetEndpointConfiguration` attributes	**Usage in `DoIpInstantiation`**	**Usage in `DltLogChannel`**
`tcpPort`	Optional	Optional
`udpPort`	Optional	Optional
`ipv4MulticastIpAddress`	N/A	N/A
`ipv6MulticastIpAddress`	N/A	N/A
`communicationConnector`	Mandatory	Mandatory

Table 9.45: Allowed usage of `PlatformModuleEthernetEndpointConfiguration` attributes

9.6 Log and Trace module configuration

The Log and Trace functionality in AUTOSAR supports the monitoring of applications and provides means to forward logging information onto the communication bus, the console, or to the file system.

The logging information is put into a standardized delivery and presentation format that is described in more detail in the Log and Trace Protocol specification [23].

The format contains meta-data that identifies for example the application that produces the logging information.

This chapter describes settings that are available in the Machine Manifest to configure the logging framework. Some of these settings will be put as meta-data into the Log and Trace messages.

[TPS_MANI_03162]{DRAFT} **`Machine`-specific configuration settings for the Log and Trace functional cluster** ⌈The `Machine`-specific configuration settings for the Log and Trace functional cluster are collected in `LogAndTraceInstantiation`.⌋ *(RS_MANI_00023)*

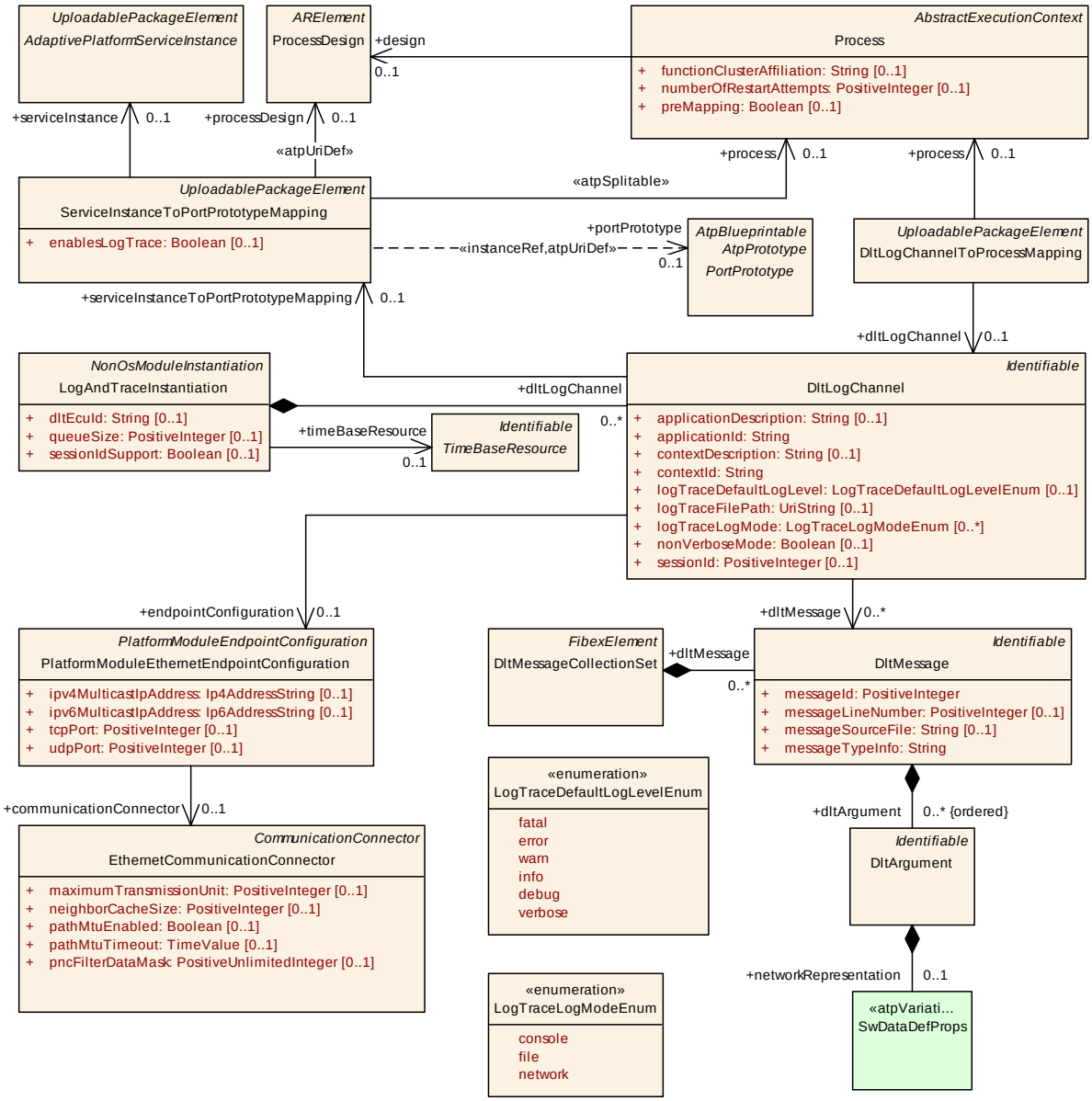


Figure 9.23: Log and Trace module configuration

Class	**LogAndTraceInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::LogAndTrace			
Note	This meta-class defines the attributes for the Log&Trace configuration on a specific machine. **Tags:**atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dltEcuId	String	0..1	attr	This attribute defines the name of the ECU for use within the Dlt protocol.

△

Class	**LogAndTraceInstantiation**			
dltLogChannel	DltLogChannel	*	aggr	DltLogChannels that are configured for the log/trace message output **Tags:**atp.Status=draft
queueSize	PositiveInteger	0..1	attr	Length of the queue (in which messages can be stored before processing) in the unit "Log message".
sessionId Support	Boolean	0..1	attr	This attribute defines whether the sessionId is used or not.
timeBase Resource	TimeBaseResource	0..1	ref	This reference is used to describe to which time base the Log and Trace module has access. From the Time Base Resource the Log and Trace module gets the needed information to generate the time stamp. **Tags:**atp.Status=draft

Table 9.46: LogAndTraceInstantiation

[TPS_MANI_03274]{DRAFT} **Configuration of log and trace message source** ⌈The `DltLogChannel` is used to to configure the log and trace message for a source defined by

- the `applicationId` that identifies the application that the log and trace message originates. The relationship to the `Executable` is created by the `DltLogChannelToProcessMapping` that maps the `DltLogChannel` to a `Process` that in turn refers the `Executable` in the `executable` role.
- the `contextId` that defines a user defined ID to group Log and Trace Messages that are generated by an application. There is the option to define the `contextId` for a specific `AdaptivePlatformServiceInstance` that is provided or consumed by the application and in this case the relationship is created by the `DltLogChannel.serviceInstanceToPortPrototypeMapping` that maps a `AdaptivePlatformServiceInstance` to a `PortPrototype` in the context of a `Process`. Please note that the `contextId` may also be a fixed number (or a set of fixed numbers) for a functional cluster. In such a case the `DltLogChannel.serviceInstanceToPortPrototypeMapping` is not used.
- the `sessionId` that identifies the instance in case the application is instantiated several times. The relationship to the `Process` is created by the `DltLogChannelToProcessMapping` that maps the `DltLogChannel` to the `Process` in case that the same `Executable` is supposed to be executed in several instances (i.e. in the form of POSIX processes) on the same platform.

⌋*(RS_MANI_00023)*

[constr_5243]{DRAFT} **Restriction of `LogAndTraceInstantiation.dltEcuId` attribute value** ⌈The `LogAndTraceInstantiation.dltEcuId` attribute value shall be composed of four ASCII characters.⌋*()*

[TPS_MANI_03160]{DRAFT} **Further configuration options in `DltLogChannel`** ⌈

- The `applicationDescription` is an optional setting that allows to describe the `applicationId` as descriptive text.

- The contextDescription is an optional setting that allows to describe the contextId as descriptive text.
- logTraceLogMode defines the destination(s) to which the log messages will be forwarded.
- logTraceDefaultLogLevel defines the initial log reporting level for the application instance. The log level defines the severity grade of the Log Message.
- logTraceFilePath defines the destination file to which the logging information is passed.

⌋(*RS_MANI_00037*)

[TPS_MANI_01272]{DRAFT} **Duplicate entries in logTraceLogMode** ⌈Duplicate entries (ie. set to the same value within the collection defined by LogTraceLogModeEnum) in the attribute logTraceLogMode shall be ignored.⌋(*RS_MANI_00037*)

[constr_3426]{DRAFT} **The logTraceFilePath is mandatory in case that logTraceLogMode is set to file** ⌈If one in the collection of logTraceLogMode is set to file the logTraceFilePath shall be set to a value.⌋()

[constr_3427]{DRAFT} **The logTraceFilePath is only relevant if logTraceLogMode is set to file** ⌈The logTraceFilePath shall only be used if one of the collection of logTraceLogMode is set to file.⌋()

Class	**DltLogChannel**			
Package	M2::AUTOSARTemplates::SystemTemplate::Dlt			
Note	This element contains the settings for the log/trace message output for a tuple of ApplicationId and ContextId (verbose mode) or a SessionId (non-verbose mode).			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
applicationDescription	String	0..1	attr	This attribute can be used to describe the applicationId that is used in the log and trace message in more detail.
applicationId	String	1	attr	This attribute identifies the SW-C/BSW module in the log and trace message.
contextDescription	String	0..1	attr	This attribute can be used to describe the contextId that is used in the log and trace message in more detail.
contextId	String	1	attr	This attribute is used to group log and trace messages produced by a SW-C/BSW modules to distinguish functionality (representing e.g. a library of the adaptive foundation linked into the application).
dltLogChannelDesign	DltLogChannelDesign	0..1	ref	This reference represents the identification of the design-time representation for the DltLogChannel that owns the reference. **Tags:**atp.Status=draft
dltMessage	DltMessage	*	ref	Reference to DltMessages that can be transported over the DltLogChannel in the DltPdu.
endpointConfiguration	PlatformModuleEthernetEndpointConfiguration	0..1	ref	Network configuration (Protocol, Port, IP Address) for transmission of dlt messages on a specific VLAN. **Tags:**atp.Status=draft

▽

△

Class	**DltLogChannel**			
logTraceDefaultLogLevel	LogTraceDefaultLogLevelEnum	0..1	attr	This attribute allows to set the initial log reporting level for a logTraceProcessId (ApplicationId). **Tags:**atp.Status=draft
logTraceFilePath	UriString	0..1	attr	This attribute defines the destination file to which the logging information is passed. **Tags:**atp.Status=draft
logTraceLogMode	LogTraceLogModeEnum	*	attr	This attribute defines the destination of log messages provided by the process. **Tags:**atp.Status=draft
nonVerboseMode	Boolean	0..1	attr	This attribute defines whether this channel supports non-Verbose Dlt messages. If disabled only verbose mode messages shall be used. **Tags:**atp.Status=draft
serviceInstanceToPortPrototypeMapping	ServiceInstanceToPortPrototypeMapping	0..1	ref	Optional reference to a PortPrototype of the monitored Application in case that the communication over this port is monitored and defines the ContextId. **Tags:**atp.Status=draft
sessionId	PositiveInteger	0..1	attr	This attribute allows distinguishing log/trace messages from different instances of the same SW-C. It is required if sessionIdSupport of the aggregating DltConfig is True.

Table 9.47: DltLogChannel

Enumeration	**LogTraceLogModeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::LogAndTrace
Note	This enum defines the possible destinations of a log&trace message. **Tags:**atp.Status=draft
Literal	*Description*
console	Destination of log message will be the console output. **Tags:**atp.EnumerationLiteralIndex=0
file	Destination of log message will be a file on the file system. **Tags:**atp.EnumerationLiteralIndex=1
network	Log message will be transmitted over the communication bus. **Tags:**atp.EnumerationLiteralIndex=2

Table 9.48: LogTraceLogModeEnum

Enumeration	**LogTraceDefaultLogLevelEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::LogAndTrace
Note	This enum defines available log&trace log levels that may be used to define the severity level of a log message. **Tags:**atp.Status=draft
Literal	*Description*
debug	Detailed information for programmers **Tags:**atp.EnumerationLiteralIndex=4

▽

△

Enumeration	**LogTraceDefaultLogLevelEnum**
error	Error with impact to correct functionality **Tags:**atp.EnumerationLiteralIndex=1
fatal	Fatal error **Tags:**atp.EnumerationLiteralIndex=0
info	High level information **Tags:**atp.EnumerationLiteralIndex=3
verbose	Verbose debug message **Tags:**atp.EnumerationLiteralIndex=5
warn	Warning if correct behavior cannot be ensured **Tags:**atp.EnumerationLiteralIndex=2

Table 9.49: LogTraceDefaultLogLevelEnum

Class	**DltLogChannelToProcessMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::LogAndTrace			
Note	This meta-class represents the ability to assign a Log&Trace Channel to a Process. **Tags:** atp.Status=draft atp.recommendedPackage=DltLogChannelToProcessMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dltLogChannel	DltLogChannel	0..1	ref	Reference to the Log&Trace channel that contains the settings for the log/trace message output for a tuple of ApplicationId and ContextId (verbose mode) or a Session Id (non-verbose mode). **Tags:**atp.Status=draft
process	Process	0..1	ref	Reference to the Process that is monitored by the DltLog Channel. **Tags:**atp.Status=draft

Table 9.50: DltLogChannelToProcessMapping

A Log message may be sent in two modes: either VERBOSE or NON-VERBOSE mode.

In the VERBOSE mode the payload of the log message contains values together with type information such that the external client can interpret the received data.

In NON-VERBOSE mode the payload of the log message contains a unique message ID and some data. The description of the payload layout according to the corresponding `messageId` is described by the `DltMessage` element.

Class	DltMessageCollectionSet			
Package	M2::AUTOSARTemplates::SystemTemplate::Dlt			
Note	Collection of DltMessages **Tags:**atp.recommendedPackage=DltMessageCollectionSets			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dltMessage	DltMessage	*	aggr	Collection of DltMessages in the DltMessageCollection Set.

Table 9.51: DltMessageCollectionSet

Class	DltMessage			
Package	M2::AUTOSARTemplates::SystemTemplate::Dlt			
Note	This element defines a DltMessage.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dltArgument (ordered)	DltArgument	*	aggr	Ordered collection of DltArguments in the DltMessage.
messageId	PositiveInteger	1	attr	This attribute defines the unique Id for the DltMessage.
messageLine Number	PositiveInteger	0..1	attr	This attribute describes the position in the source file in which this log message was called.
messageSource File	String	0..1	attr	This attribute describes the source file in which this log message was called.
messageType Info	String	1	attr	This attribute describes the message Type

Table 9.52: DltMessage

Class	DltArgument			
Package	M2::AUTOSARTemplates::SystemTemplate::Dlt			
Note	This element defines an Argument in a DltMessage.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
network Representation	SwDataDefProps	0..1	aggr	Definition of the networkRepresentation of the Dlt Argument.

Table 9.53: DltArgument

Please note that the allowed usage of `DltArgument.networkRepresentation` is described in the System Template [17] in [constr_5098]. The allowed usage of `DltMessage.messageTypeInfo` is described in the same document by [constr_5099]. Both constraints are also valid for this specification.

[TPS_MANI_03163]{DRAFT} **Network configuration for Log and Trace messages**
⌈The output channel on Ethernet for Log and Trace messages is configured with the `PlatformModuleEthernetEndpointConfiguration` that is referenced by the `DltLogChannel` in the role `endpointConfiguration`. The attributes `tcpPort` and `udpPort` are used to configure the Transport Protocol (Udp or Tcp) and the used

Port number. The IP Address is configured in the NetworkEndpoint that is referenced by the PlatformModuleEthernetEndpointConfiguration via the EthernetCommunicationConnector.⌋(RS_MANI_00023)

9.7 Network Management configuration

[TPS_MANI_03166]{DRAFT} **Machine-specific configuration settings for NM module** ⌈The Machine-specific configuration settings for Nm are collected in NmInstantiation.⌋(RS_MANI_00023)

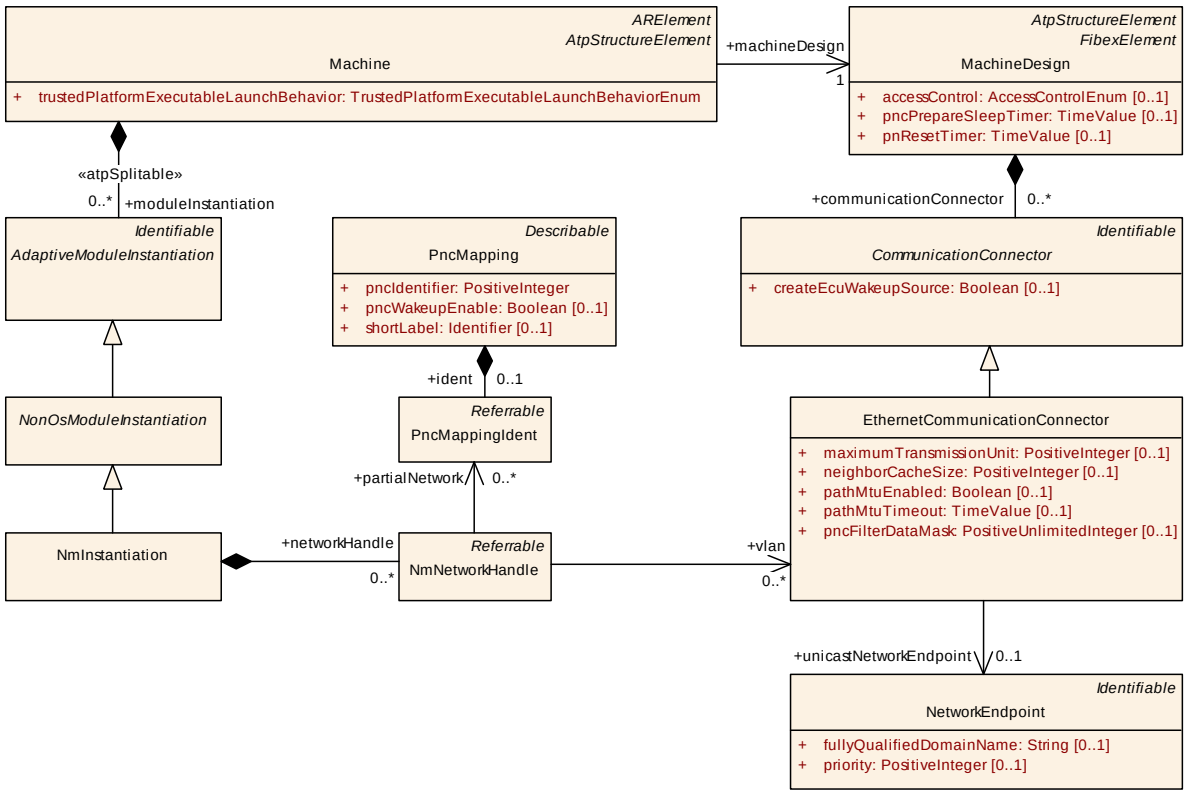


Figure 9.24: Network configuration for Nm

Class	**NmInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class defines the attributes for the Nm configuration on a specific machine. **Tags:**atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	NmInstantiation			
networkHandle	NmNetworkHandle	*	aggr	Supported NmNetworkHandles used to control Partial Network Clusters/VLANs. **Tags:**atp.Status=draft

Table 9.54: NmInstantiation

Class	NmNetworkHandle			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	Group of partialNetworks and/or VLANs that can be controlled collectively. **Tags:**atp.Status=draft			
Base	*ARObject*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
partialNetwork	PncMappingIdent	*	ref	Reference to a Partial Network that is included in the Nm NetworkHandle. **Tags:**atp.Status=draft
vlan	EthernetCommunication Connector	*	ref	Reference to a VLAN that is included in the NmNetwork Handle. **Tags:**atp.Status=draft

Table 9.55: NmNetworkHandle

[TPS_MANI_03226]{DRAFT} **Collection of partialNetworks and vlans in `NmNetworkHandle`** ⌈The `NmNetworkHandle` element is used to describe a collection of `partialNetwork`s and `vlan`s that can be controlled collectively by the State Management.⌋ *(RS_MANI_00023)*

The `UdpNmCluster` with all included `UdpNmNode`s is described in the `System` design model. With the reference `NmNode.machine` the relation between the `System` design model and the `NmInstantiation` on a `Machine` is established.

Typically, the `System` design model is provided by an OEM that defines the network configuration and provides all configuration settings that are relevant for a network management cluster to an integrator. The NM configuration options that will typically be set by an Integrator are collected in the `NmInstantiation` element. The Machine Manifest delivery to configure UdpNm consists of both, the `NmInstantiation` settings together with the `UdpNmCluster` and `UdpNmNode` settings.

The `NmConfig` element is a wrapper that contains all network management specific configuration settings in the `System` model.

AUTOSAR Adaptive Network Management is based on periodic NM messages, which are received by all `UdpNmNode`s in the `UdpNmCluster` via multicast. Reception of NM packets indicates that sending `UdpNmNode`s want to keep the `UdpNmCluster` awake. If any node is ready to go to sleep mode, it stops sending NM messages, but as long as NM packets from other `UdpNmNode`s are received, it postpones transition to sleep mode.

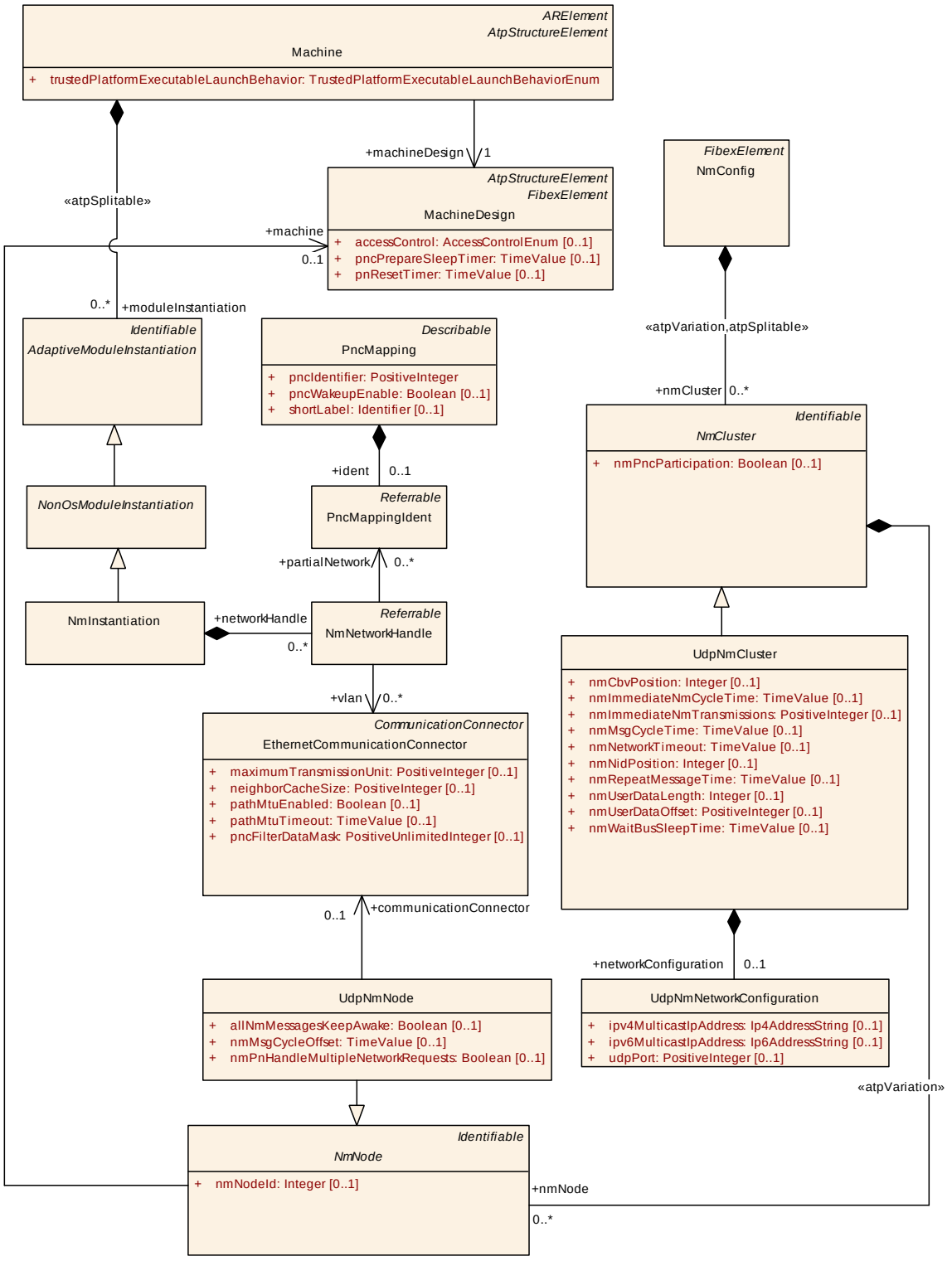


Figure 9.25: NM Cluster configuration

[TPS_MANI_03167]{DRAFT} **Network configuration for Nm** ⌈The UDP multicast connection over which Network Management messages are transported is configured with the `UdpNmNetworkConfiguration` that is aggregated by the `UdpNmCluster` in the role `networkConfiguration`. The attribute `udpPort` is used

to configure the port number over which the Nm message is transmitted and received. The IP Address is configured either by ipv4MulticastIpAddress or ipv6MulticastIpAddress.⌋*(RS_MANI_00023)*

[constr_3419]{DRAFT} **Allowed usage of UdpNmNetworkConfiguration attributes** ⌈The UdpNmNetworkConfiguration that is aggregated by UdpNmCluster in the role networkConfiguration shall have either

- ipv4MulticastIpAddress or
- ipv6MulticastIpAddress.

⌋*()*

Class	**NmConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement			
Note	Contains the all configuration elements for AUTOSAR Nm. **Tags:**atp.recommendedPackage=NmConfigs			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
nmCluster	NmCluster	*	aggr	Collection of NM Clusters atpVariation: Derived, because cluster can be variable. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=nmCluster.shortName, nmCluster.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table 9.56: NmConfig

Class	***NmCluster*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement			
Note	Set of NM nodes coordinated with use of the NM algorithm.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	CanNmCluster, FlexrayNmCluster, J1939NmCluster, UdpNmCluster			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communicationCluster	CommunicationCluster	0..1	ref	Association to a CommunicationCluster in the topology description.
nmNode	NmNode	*	aggr	Collection of NmNodes of the NmCluster. atpVariation: Derived, because NmNode can be variable. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
nmPncParticipation	Boolean	0..1	attr	Defines whether this NmCluster contributes to the partial network mechanism.

Table 9.57: NmCluster

Class	**UdpNmCluster**			
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement			
Note	Udp specific NmCluster attributes			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *NmCluster*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
network Configuration	UdpNmNetwork Configuration	0..1	aggr	Configuration of a UDP port and UDP multicast IP address of the Nm communication on a VLAN. **Tags:**atp.Status=draft
nmCbvPosition	Integer	0..1	attr	Defines the position of the control bit vector within the Nm Pdu (Byte position). If this attribute is not configured, the Control Bit Vector is not used.
nmImmediate NmCycleTime	TimeValue	0..1	attr	Defines the immediate NmPdu cycle time in seconds which is used for nmImmediateNmTransmissions NmPdu transmissions. This attribute is only valid if nmImmediate NmTransmissions is greater one.
nmImmediate Nm Transmissions	PositiveInteger	0..1	attr	Defines the number of immediate NmPdus which shall be transmitted. If the value is zero no immediate NmPdus are transmitted. The cycle time of immediate NmPdus is defined by nmImmediateNmCycleTime.
nmMsgCycle Time	TimeValue	0..1	attr	Period of a NmPdu in seconds. It determines the periodic rate in the periodic transmission mode with bus load reduction and is the basis for transmit scheduling in the periodic transmission mode without bus load reduction.
nmNetwork Timeout	TimeValue	0..1	attr	Network Timeout for NmPdus in seconds. It denotes the time how long the UdpNm shall stay in the Network Mode before transition into Prepare Bus-Sleep Mode shall take place.
nmNidPosition	Integer	0..1	attr	Defines the byte position of the source node identifier within the NmPdu. If this attribute is not configured, the Node Identification is not used.
nmRepeat MessageTime	TimeValue	0..1	attr	Timeout for Repeat Message State in seconds. Defines the time how long the NM shall stay in the Repeat Message State.
nmUserData Length	Integer	0..1	attr	Defines the length in bytes of the user data contained in the Nm message. User data excludes the PN information.
nmUserData Offset	PositiveInteger	0..1	attr	Specifies the offset (in bytes) of the user data information in the NM message. User data excludes the PN information. **Tags:**atp.Status=draft
nmWaitBus SleepTime	TimeValue	0..1	attr	Timeout for bus calm down phase in seconds. It denotes the time how long the CanNm shall stay in the Prepare Bus-Sleep Mode before transition into Bus-Sleep Mode shall take place.
vlan	EthernetPhysical Channel	0..1	ref	Reference to the vlan (represented by the Ethernet PhysicalChannel) this UdpNmCluster shall apply to.

Table 9.58: UdpNmCluster

Class	***NmNode*** (abstract)
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement
Note	The linking of NmEcus to NmClusters is realized via the NmNodes.
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

△

Class	***NmNode*** (abstract)			
Subclasses	CanNmNode, FlexrayNmNode, J1939NmNode, UdpNmNode			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
machine	MachineDesign	0..1	ref	Reference to the machine that contains the NmNode. **Tags:**atp.Status=draft
nmNodeId	Integer	0..1	attr	Node identifier of local NmNode. Shall be unique in the NmCluster.

Table 9.59: NmNode

Class	**UdpNmNode**			
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement			
Note	Udp specific NM Node attributes.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *NmNode*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allNmMessages KeepAwake	Boolean	0..1	attr	Specifies if Nm drops irrelevant NM PDUs. false: Only NM PDUs with a Partial Network Information Bit (PNI) = true and containing a Partial Network request for this ECU trigger the standard RX indication handling and thus keep the ECU awake true: Every NM PDU triggers the standard RX indication handling and keeps the ECU awake
communication Connector	EthernetCommunication Connector	0..1	ref	Reference to the CommunicationConnector that represents the UdpNmNode in the topology description. **Tags:**atp.Status=draft
nmMsgCycle Offset	TimeValue	0..1	attr	Node specific time offset in the periodic transmission node. It determines the start delay of the transmission. Specified in seconds.
nmPnHandle MultipleNetwork Requests	Boolean	0..1	attr	Specifies if NM performs an additional transition from Network Mode to Repeat Message State (true) or not (false).

Table 9.60: UdpNmNode

9.7.1 UdpNm configuration constraints

Please note that the Classic Platform and the Adaptive Platform are using the same model for configuration of UdpNm. Some Classic Platform features like the NmCoordinator and UdpNmClusterCoupling are not supported in Adaptive Autosar.

But the TPS_SystemTemplate [17] contains a more detailed description of `UdpNmClusters` and `UdpNmNodes` and defines modeling constraints that are also valid for the Adaptive Platform.

The following constraints of the TPS_SystemTemplate [17] shall be considered if a `UdpNmCluster` with `UdpNmNodes` is described:

- [constr_3041]
- [constr_3042]

- [constr_3078]
- [constr_3079]
- [constr_3080]
- [constr_5223]
- [constr_5224]
- [constr_5225]
- [constr_5226]

In addition the following Adaptive Platform specific constraints are valid:

[constr_5227]{DRAFT} **Mandatory elements of `UdpNmCluster`** ⌈The following attributes shall always be defined for the `UdpNmCluster`:

- `nmMsgCycleTime`
- `nmNetworkTimeout`
- `nmRepeatMessageTime`
- `nmWaitBusSleepTime`
- `communicationCluster`

⌋()

[constr_5228]{DRAFT} **Partial Networking timing constraint** ⌈For Partial Networking the following timing constraints shall be ensured: (`MachineDesign.pnResetTimer` + `MachineDesign.pncPrepareSleepTimer`) < `UdpNmCluster.nmNetworkTimeout`⌋()

9.8 Update and Configuration Management

[TPS_MANI_01226]{DRAFT} **Machine-specific configuration settings for the UCM module** ⌈The Machine-specific configuration settings for Ucm are collected in meta-class `UcmModuleInstantiation`.⌋(*RS_MANI_00023*)

Class	**UcmModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Ucm			
Note	This meta-class represents the ability to define a definition of a UCM instantiation. **Tags:**atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	UcmModuleInstantiation			
identifier	String	1	attr	This represents the identification of a UCM.
maxNumberOf Parallel Transfers	PositiveInteger	0..1	attr	This attribute supports the configuration of the maximum number of parallel transfers that the Ucm on the enclosing Machine is allowed to create.

Table 9.61: UcmModuleInstantiation

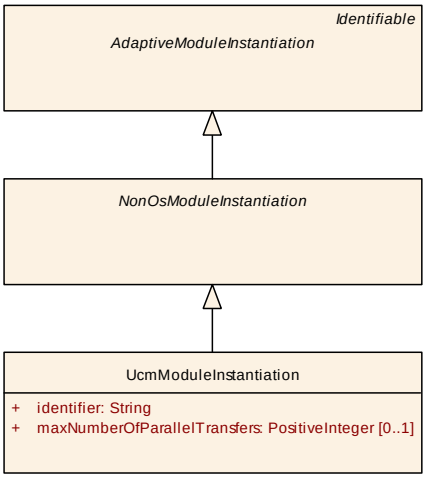


Figure 9.26: Modeling of `UcmModuleInstantiation`

[TPS_MANI_01227]{DRAFT} **Semantics of attribute `UcmModuleInstantiation.identifier`** ⌈Attribute `UcmModuleInstantiation.identifier` shall be used to identify a specific Ucm on a specific `Machine` during a service discovery run by a master UCM or VUM.⌋ *(RS_MANI_00023)*

The usage of attribute `UcmModuleInstantiation.identifier` is documented in Figure 9.27. The master UCM or VUM acts as a client in a service discovery that is configured to search for *any* server.

The individual UCMs offer their service and then the master UCM as the client calls a specific `method` in the server's `ServiceInterface` to reveal the `identifier` of each server.

In the case of this example there are three slave UCMs with `identifier` set to 1, 2, and 3. The master UCM instantiates a proxy for each of the slave UCMs such that the value of the respective `identifier` can be retrieved from the proxy in order to be able to communicate with a specific slave UCM.

The master UCM or VUM can then instantiate proxies for each service offer and programmatically access the respective server going forward.

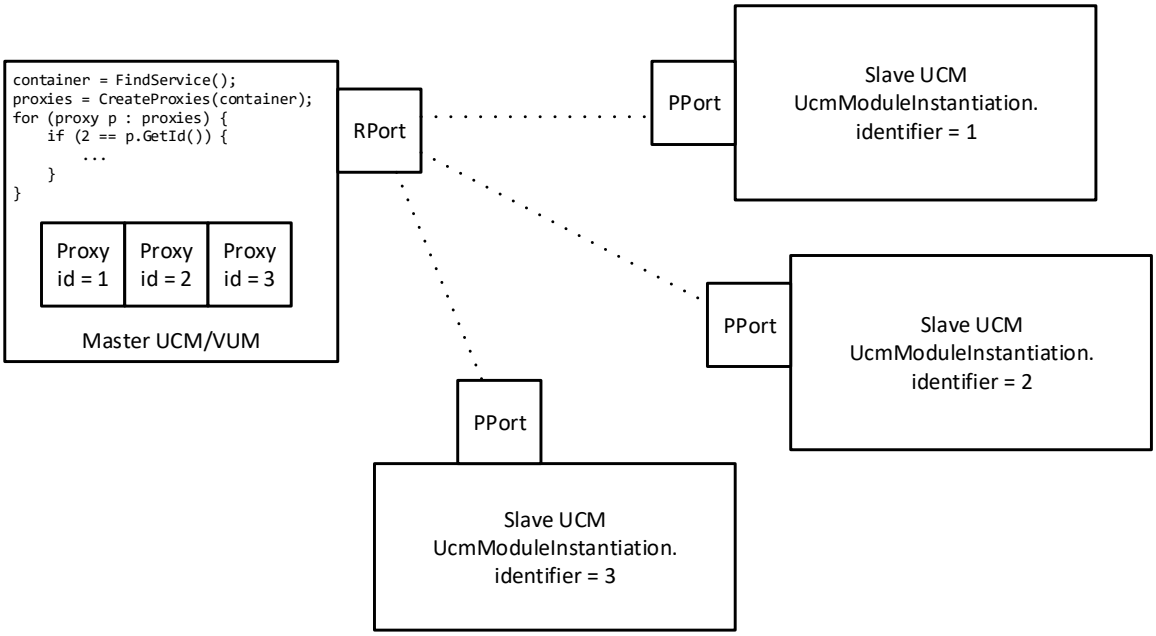


Figure 9.27: Identification of slave UCM modules by the master UCM

[constr_1691]{DRAFT} **UcmModuleInstantiation.identifier shall be unique**
⌈The value of attribute UcmModuleInstantiation.identifier shall be unique for each Machine in a given vehicle.⌋*()*

9.9 IAM configuration

The definition of the deployment for the *Identity and Access Manager* represents the creation of actual grants, as opposed to the definition of grants on design level.

One important aspect of the modeling on deployment level is that it is not intended to include a large portion of the design model. The goal is to keep the deployment part as self-contained as possible.

While this approach represents a significant benefit for the size of deployment models it also creates some sort of disconnect between design and deployment. In other words, the connection of the modeling of a specific Grant to the respective intent in the design model is not immediately obvious.

To mitigate this issue, AUTOSAR introduced the GrantDesign that in turn allows for the identification of the corresponding intent modeling. When loading the design model and deployment model together into a suitable tool it would still be possible to run an analysis in terms of completeness of the overall IAM configuration.

The enforcement of access restrictions is not mandatory for a Machine. Therefore, the existence of a Grant by itself is not sufficient to activate the IAM mechanisms.

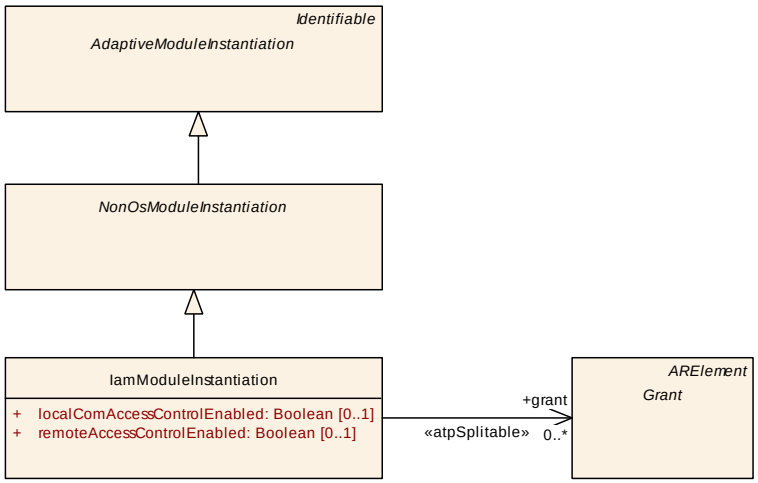


Figure 9.28: Modeling of the `IamModuleInstantiation`

[constr_1695]{DRAFT} **Semantics of a `Grant` depends on the existence of `IamModuleInstantiation`** ⌈The existence of `Grant`s shall only be enforced if in the context of the enclosing `Machine` an `IamModuleInstantiation` has been defined and is referencing the `Grant`.⌋*()*

9.9.1 Com Grant Deployment

[TPS_MANI_01237]{DRAFT} **Semantics of meta-class `ComFieldGrant`** ⌈Meta-class `ComFieldGrant` shall be used to award access to a given `field` (identified by means of the reference to meta-class `ServiceFieldDeployment` in the role `serviceDeployment`) in the context of a given `AdaptivePlatformServiceInstance` referenced in the role `serviceInstance`.⌋*(RS_MANI_00060)*

In other words, if a given `AdaptivePlatformServiceInstance` and the respective `ServiceFieldDeployment` are not referenced from a `ComFieldGrant` and an `IamModuleInstantiation` exists **then this specific communication shall be suppressed**.

Class	**ComFieldGrant**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to grant access to a ServiceInterface.field. **Tags:** atp.Status=draft atp.recommendedPackage=Grants			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrant*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ComFieldGrant**			
design	ComFieldGrantDesign	0..1	ref	This reference identifies the ComFieldGrantDesign that the enclosing ComFieldGrant was created from. **Stereotypes:** atpUriDef **Tags:**atp.Status=draft
role	FieldAccessEnum	1	attr	This attribute provides the ability to further specify the access to the ServiceInterface.field.
service Deployment	ServiceField Deployment	1	ref	This reference identifies the applicable deployment within the context of an AdaptivePlatformServiceInstance for which the grant applies. **Tags:**atp.Status=draft

Table 9.62: ComFieldGrant

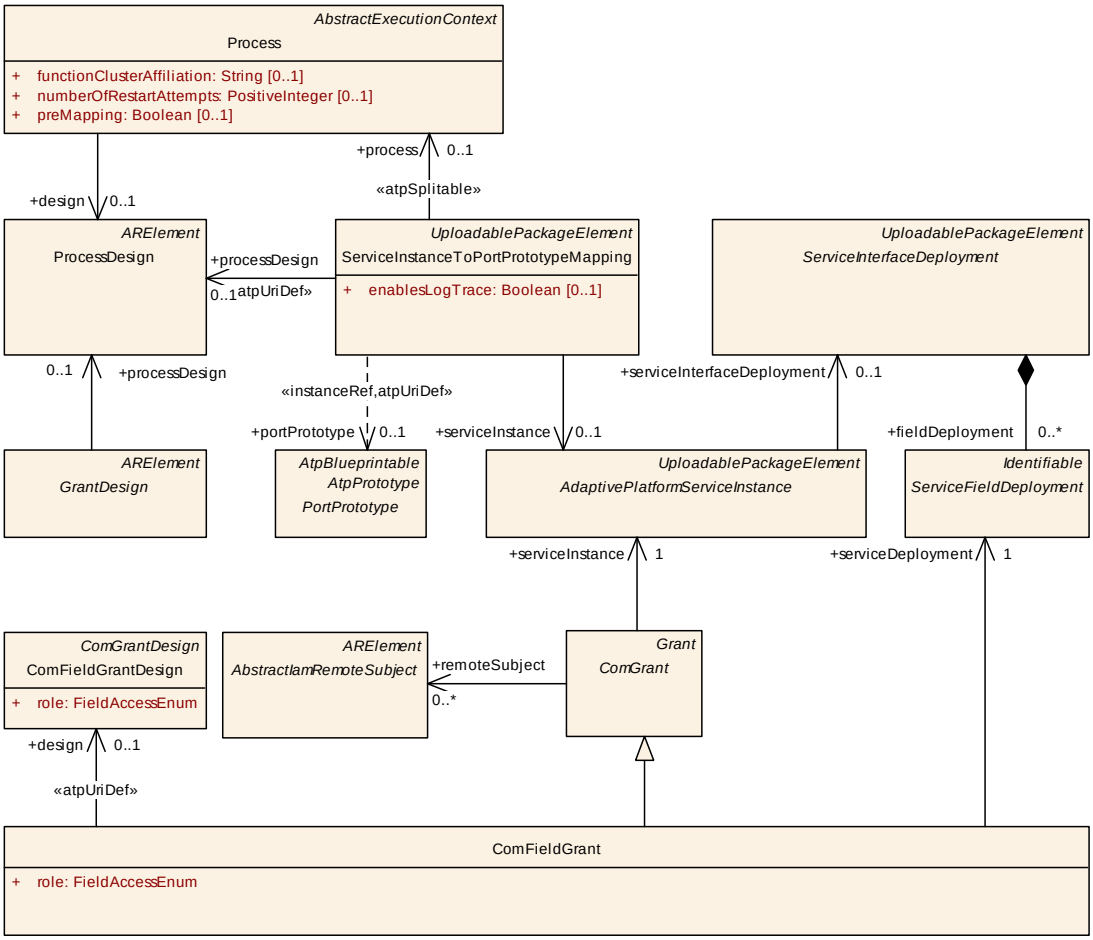


Figure 9.29: Modeling of the `ComFieldGrant`

[TPS_MANI_01238]{DRAFT} **Semantics of meta-class `ComMethodGrant`** ⌈Meta-class `ComMethodGrant` shall be used to clear the call of a given `method` (identified by means of the reference to meta-class `ServiceMethodDeployment` in the role `serviceDeployment`) in the context of a given `AdaptivePlatformServiceInstance` referenced in the role `serviceInstance`.⌋*(RS_MANI_00060)*

In other words, if a given `AdaptivePlatformServiceInstance` and the respective `ServiceMethodDeployment` are not referenced from a `ComMethodGrant` and

an `IamModuleInstantiation` exists **then this specific communication shall be suppressed**.

Class	**ComMethodGrant**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to grant access to a ServiceInterface.method. **Tags:** atp.Status=draft atp.recommendedPackage=Grants			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrant*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ComMethodGrant Design	0..1	ref	This reference identifies the ComMethodGrantDesign that the enclosing ComMethodGrant was created from. **Stereotypes:** atpUriDef **Tags:**atp.Status=draft
service Deployment	ServiceMethod Deployment	1	ref	This reference identifies the applicable deployment within the context of an AdaptivePlatformServiceInstance for which the grant applies. **Tags:**atp.Status=draft

Table 9.63: ComMethodGrant

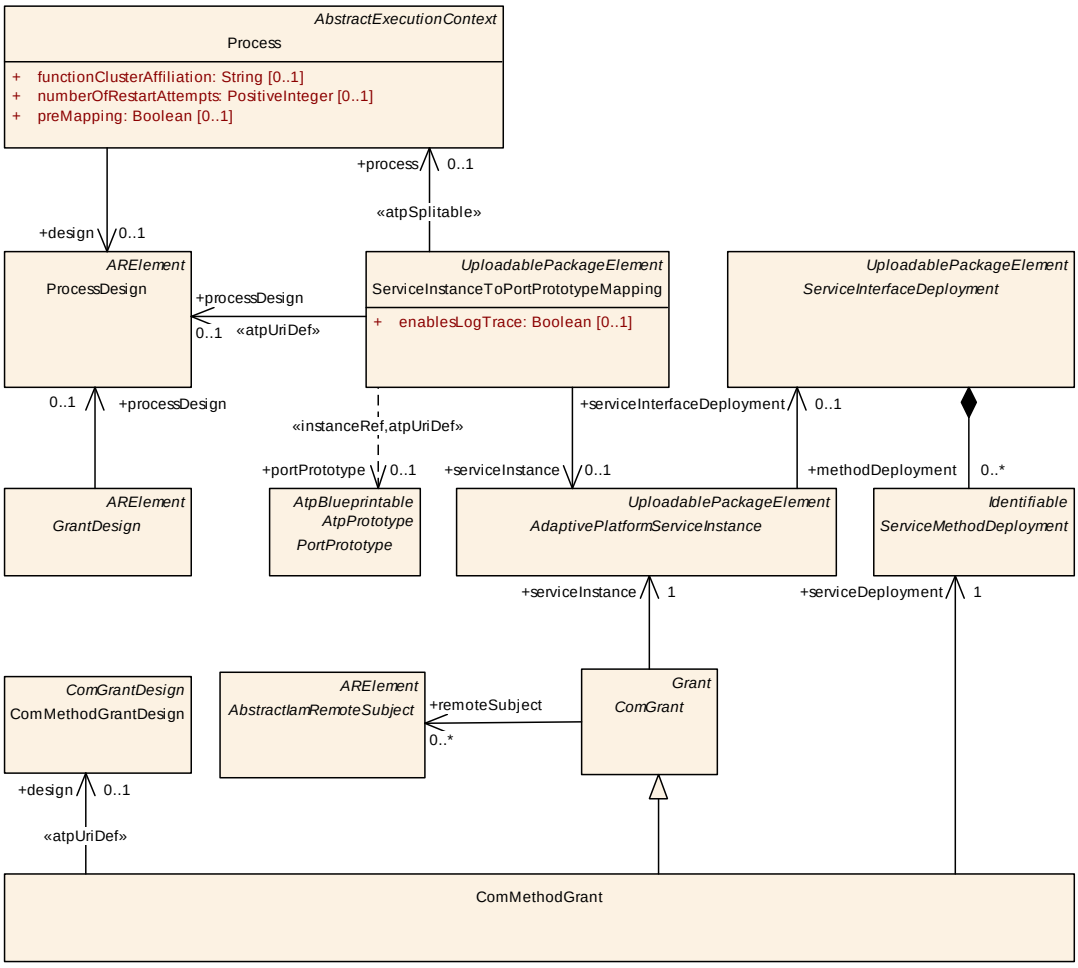


Figure 9.30: Modeling of the `ComMethodGrant`

[TPS_MANI_01239]{DRAFT} **Semantics of meta-class `ComEventGrant`** ⌈Meta-class `ComEventGrant` shall be used to award access to a given `event` (identified by means of the reference to meta-class `ServiceEventDeployment` in the role `serviceDeployment`) in the context of a given `AdaptivePlatformServiceInstance` referenced in the role `serviceInstance`.⌋*(RS_MANI_00060)*

In other words, if a given `AdaptivePlatformServiceInstance` and the respective `ServiceEventDeployment` are not referenced from a `ComEventGrant` and an `IamModuleInstantiation` exists **then this specific communication shall be suppressed**.

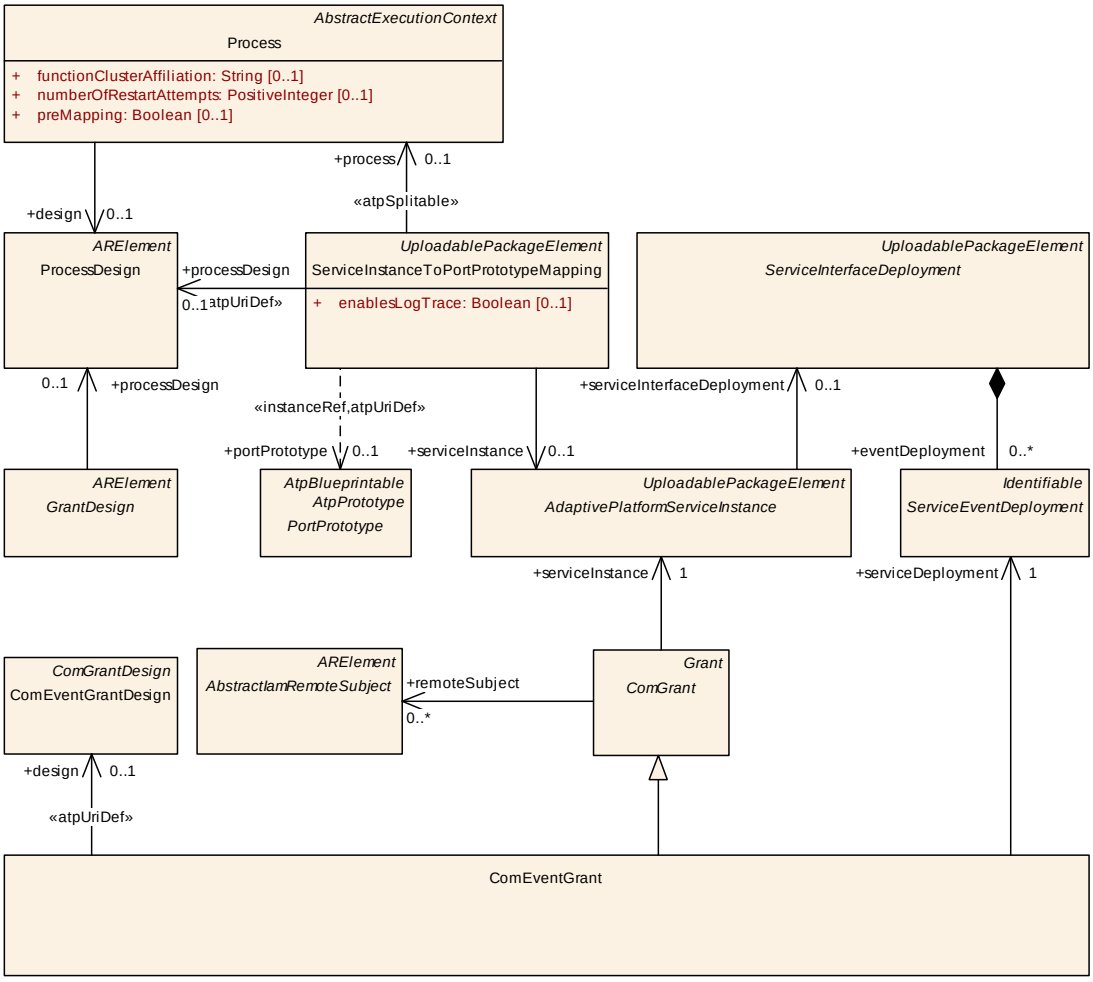


Figure 9.31: Modeling of the `ComEventGrant`

Class	**ComEventGrant**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement
Note	This meta-class represents the ability to grant access to a ServiceInterface.event. **Tags:** atp.Status=draft atp.recommendedPackage=Grants

△

Class	ComEventGrant			
Base	ARElement, ARObject, CollectableElement, ComGrant, Grant, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
design	ComEventGrantDesign	0..1	ref	This reference identifies the ComEventGrantDesign that the enclosing ComEventGrant was created from. **Stereotypes:** atpUriDef **Tags:**atp.Status=draft
service Deployment	ServiceEvent Deployment	1	ref	This reference identifies the applicable deployment within the context of an AdaptivePlatformServiceInstance for which the grant applies. **Tags:**atp.Status=draft

Table 9.64: ComEventGrant

The enforcement of service discovery rights is modeled by means of meta-classes `ComOfferServiceGrant` and `ComFindServiceGrant`.

[TPS_MANI_01240]{DRAFT} **Semantics of meta-class `ComOfferServiceGrant`**
⌈Meta-class `ComOfferServiceGrant` shall be used to award the right to offer the referenced `AdaptivePlatformServiceInstance`.⌋*(RS_MANI_00060)*

Class	ComOfferServiceGrant			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to grant the offering of a service. **Tags:** atp.Status=draft atp.recommendedPackage=Grants			
Base	ARElement, ARObject, CollectableElement, Grant, Identifiable, MultilanguageReferrable, Packageable Element, Referrable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
design	ComOfferServiceGrant Design	0..1	ref	This reference identifies the ComOfferServiceGrant Design that the enclosing ComOfferServiceGrant was created from. **Stereotypes:** atpUriDef **Tags:**atp.Status=draft
serviceInstance	AdaptivePlatform ServiceInstance	1	ref	This reference identifies the AdaptivePlatformService Instances for which the grant applies. **Tags:**atp.Status=draft

Table 9.65: ComOfferServiceGrant

[TPS_MANI_01241]{DRAFT} **Semantics of meta-class `ComFindServiceGrant`**
⌈Meta-class `ComFindServiceGrant` shall be used to award the right to start a find of the referenced `AdaptivePlatformServiceInstance`.⌋*(RS_MANI_00060)*

Class	**ComFindServiceGrant**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to grant the finding a service. **Tags:** atp.Status=draft atp.recommendedPackage=Grants			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ComFindServiceGrant Design	0..1	ref	This reference identifies the ComFindServiceGrantDesign that the enclosing ComFindServiceGrant was created from. **Stereotypes:** atpUriDef **Tags:**atp.Status=draft
serviceInstance	AdaptivePlatform ServiceInstance	0..1	ref	This reference identifies the AdaptivePlatformService Instances for which the grant applies. **Tags:**atp.Status=draft

Table 9.66: ComFindServiceGrant

9.9.2 Grant Deployment for Raw Streaming Data

The definition of abstract meta-class `RawDataStreamGrant` on the level of deployment complements the existence of `RawDataStreamGrantDesign` on design level.

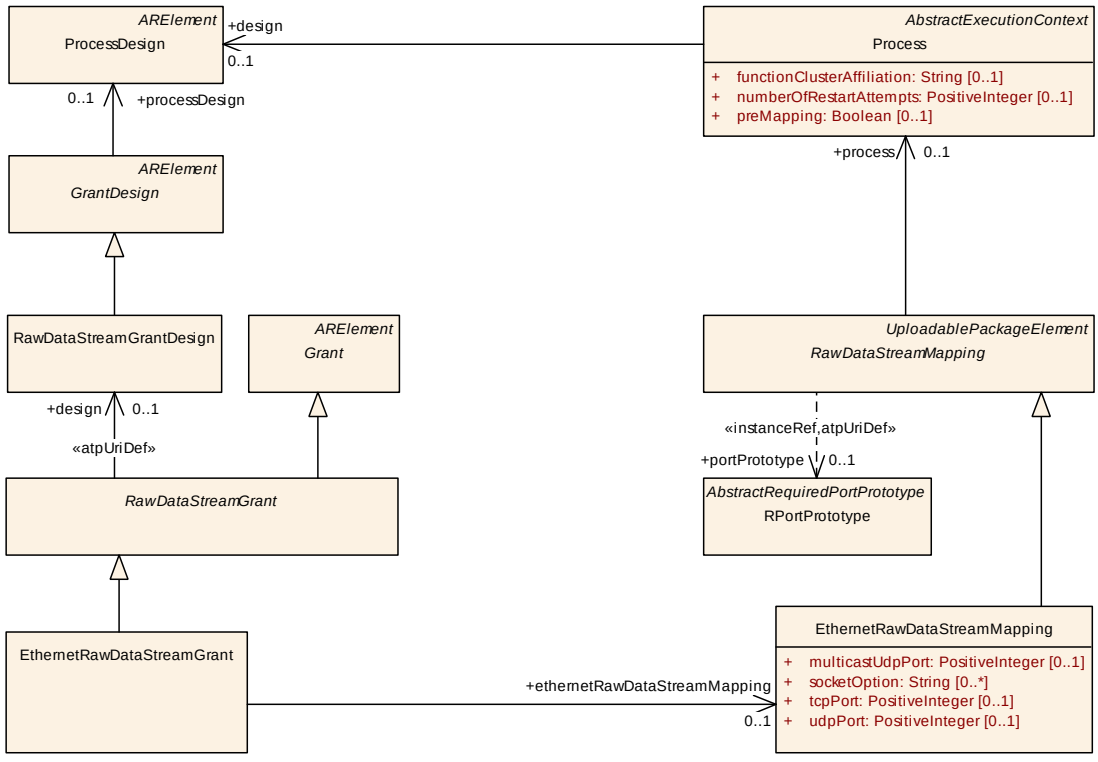


Figure 9.32: Modeling of the `RawDataStreamGrant`

[TPS_MANI_01307]{DRAFT} **Semantics of meta-class EthernetRawDataStreamGrant** ⌈Meta-class EthernetRawDataStreamGrant provides the deployment-level IAM semantics for raw data streams that run on TCP/IP sockets. For this purpose, the reference in the role ethernetRawDataStreamMapping to meta-class EthernetRawDataStreamMapping exists.⌋*()*

Class	***RawDataStreamGrant*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This abstract meta-class represents the ability to define the IAM configuration for a RawDataStream on deployment level. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	EthernetRawDataStreamGrant			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	RawDataStreamGrant Design	0..1	ref	This reference identifies the RawDataStreamGrantDesign that the enclosing RawDataStreamEventGrant was created from. **Stereotypes:** atpUriDef **Tags:**atp.Status=draft

Table 9.67: RawDataStreamGrant

Class	**EthernetRawDataStreamGrant**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to define the IAM configuration for a EthernetRawDataStream on deployment level. **Tags:** atp.Status=draft atp.recommendedPackage=Grants			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *RawDataStreamGrant*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ethernetRaw DataStream Mapping	EthernetRawData StreamMapping	0..1	ref	This reference identifies the applicable EthernetRawData Stream to which the enclosing EthernetRawDataStream Grant shall apply. **Tags:**atp.Status=draft

Table 9.68: EthernetRawDataStreamGrant

9.9.3 Remote access control

The overview of the remote access control was already introduced in chapter 3.20.3. Please note that for the modeling of AbstractIamRemoteSubjects the same approach is used in the Design and in the Deployment. So if AbstractIamRemoteSubjects were defined during the design phase they can be taken over into the deployment.

This chapter defines how ComGrants with a defined remoteSubjects shall be interpreted.

[TPS_MANI_03245]{DRAFT} **Definition of ComMethodGrant.remoteSubjects on server side** ⌈If the ComMethodGrant references a ProvidedApServiceInstance in the role serviceInstance then the ComMethodGrant.remoteSubject reference defines the remoteSubjects that are allowed to call the defined method that is referenced by the ComMethodGrant in the role serviceDeployment.⌋()

[TPS_MANI_03246]{DRAFT} **Definition of ComMethodGrant.remoteSubjects on client side** ⌈If the ComMethodGrant references a RequiredApServiceInstance in the role serviceInstance then the ComMethodGrant.remoteSubject reference defines the remoteSubjects to which a ServiceMethodDeployment.method call is allowed to be sent.⌋()

[TPS_MANI_03247]{DRAFT} **Definition of ComEventGrant.remoteSubjects on provider side** ⌈If the ComEventGrant references a ProvidedApServiceInstance in the role serviceInstance then the ComEventGrant.remoteSubject reference defines the remoteSubjects to which an ServiceEventDeployment.event is allowed to be sent.⌋()

[TPS_MANI_03248]{DRAFT} **Definition of ComEventGrant.remoteSubjects on receiver side** ⌈If the ComEventGrant references a RequiredApServiceInstance in the role serviceInstance then the ComEventGrant.remoteSubject reference defines the remoteSubjects from which an ServiceEventDeployment.event is allowed to be received.⌋()

[TPS_MANI_03249]{DRAFT} **Definition of ComFieldGrant.remoteSubjects on provider side** ⌈If the ComFieldGrant references a ProvidedApServiceInstance in the role serviceInstance then the ComFieldGrant.remoteSubject reference defines the remoteSubjects that are allowed to access the Field that is referenced by the ServiceFieldDeployment that in turn is referenced by the ComFieldGrant in the role serviceDeployment. This allows the following communication between the local Machine and the AbstractIamRemoteSubject:

- the Field notifier is allowed to be sent to remoteSubjects,
- if ComFieldGrant.role equals setter or getterSetter, then the setter call is allowed to be received from the remoteSubjects,
- if ComFieldGrant.role equals setter or getterSetter, then the setter return is allowed to be sent to remoteSubjects,
- if ComFieldGrant.role equals getter or getterSetter, then the getter call is allowed to be received from the remoteSubjects,
- if ComFieldGrant.role equals getter or getterSetter, then the getter return is allowed to be sent to remoteSubjects.

⌋()

[TPS_MANI_03250]{DRAFT} **Definition of ComFieldGrant.remoteSubjects on client side** ⌈If the ComFieldGrant references a RequiredApServiceInstance in the role serviceInstance then the ComFieldGrant.remoteSubject reference

defines the remoteSubjects that are allowed to provide the Field for access that is referenced by the ServiceFieldDeployment that in turn is referenced by the ComFieldGrant in the role serviceDeployment. This will allow the following communication between the local Machine and the AbstractIamRemoteSubject:

- the Field notifier is allowed to be received from remoteSubjects,
- if ComFieldGrant.role equals setter or getterSetter, then the setter call is allowed to be sent to the remoteSubjects,
- if ComFieldGrant.role equals setter or getterSetter, then the setter return is allowed to be received from remoteSubjects,
- if ComFieldGrant.role equals getter or getterSetter, the getter call is allowed to be sent to the remoteSubjects,
- if ComFieldGrant.role equals getter or getterSetter, the getter return is allowed to be received from remoteSubjects.

⌋()

Please note that a ComGrant with the remoteSubject reference defines that a remote access control for this ComGrant will be performed. Such a ComGrant with the remoteSubject reference does not enforce any Machine local access restrictions. To enforce local access restrictions for the same ServiceInterface element of the same AdaptivePlatformServiceInstance an additional ComGrant needs to be defined that points to the same serviceInstance and to the same ComFieldGrant.serviceDeployment or ComEventGrant.serviceDeployment or ComMethodGrant.serviceDeployment but does not contain the remoteSubject reference.

9.10 Crypto Deployment

This chapter explains the configuration of the Crypto functional cluster and the interaction of application Software with the Crypto stack [13].

[TPS_MANI_03260]{DRAFT} **Semantics of meta-class CryptoModuleInstantiation** ⌈The representation of the Crypto functional cluster [13] within one specific Machine is defined by the CryptoModuleInstantiation.⌋*(RS_MANI_00023)*

Class	**CryptoModuleInstantiation**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment
Note	This meta-class defines the configuration for the Crypto stack on a specific machine. **Tags:**atp.Status=draft
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*

▽

△

Class	**CryptoModuleInstantiation**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
certificateToKey SlotMapping	CryptoCertificateTo CryptoKeySlotMapping	*	aggr	List of CryptoCertificateToCryptoKeySlotMappings available in the CryptoStack. **Tags:**atp.Status=draft
crypto Certificate	CryptoCertificate	*	aggr	List of CryptoCertificates managed in the CryptoStack **Tags:**atp.Status=draft
cryptoProvider	CryptoProvider	*	aggr	List of CryptoProviders provided by the CryptoStack **Tags:**atp.Status=draft

Table 9.69: CryptoModuleInstantiation

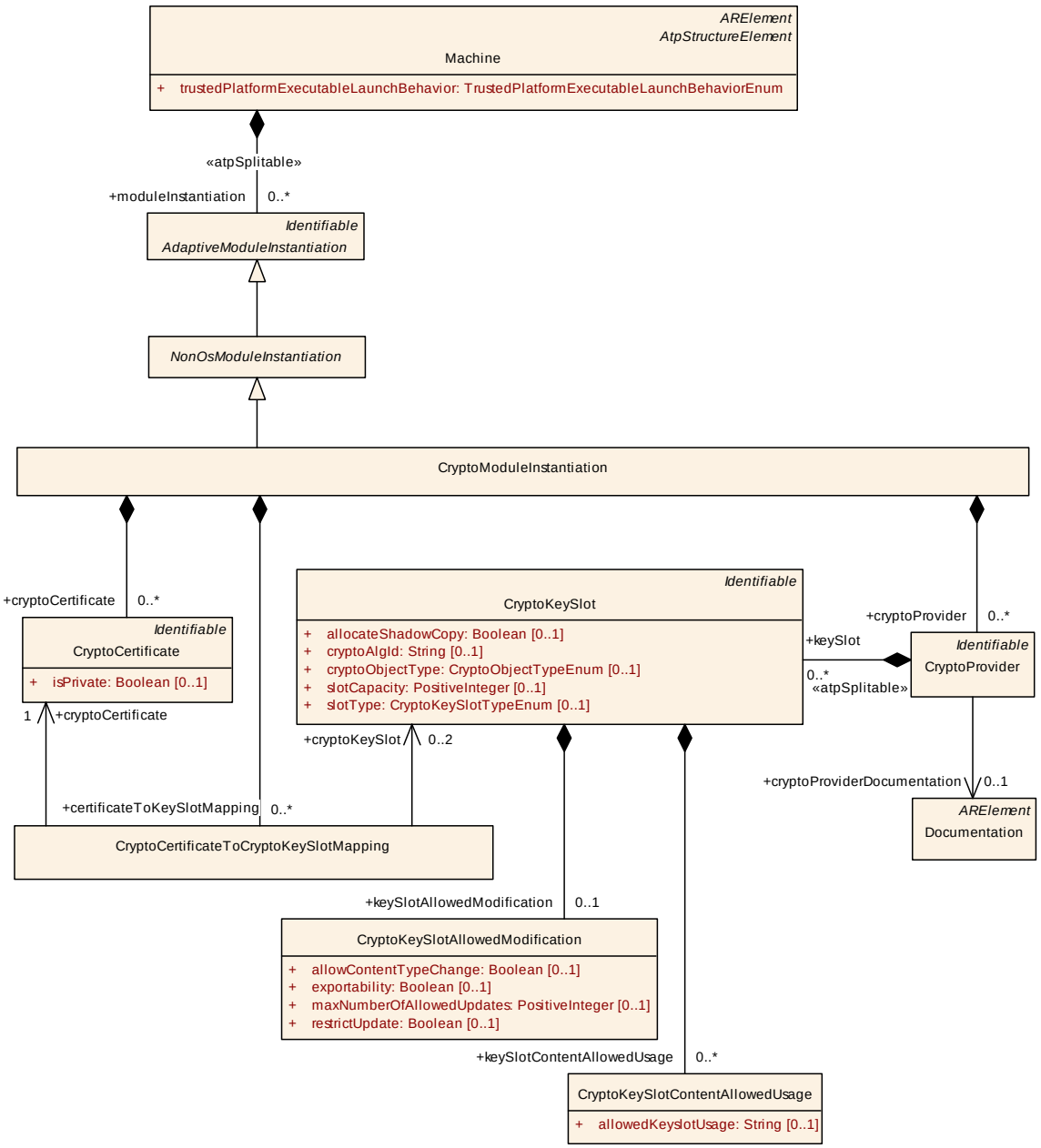


Figure 9.33: CryptoModuleInstantiation Overview

9.10.1 Crypto Provider

[TPS_MANI_03261]{DRAFT} **Support of CryptoProviders** ⌈The Crypto functional cluster is able to support multiple CryptoProviders. Each CryptoProvider implements a software- or a hardware-based cryptographic library.⌋*(RS_MANI_00023)*

Class	**CryptoProvider**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	CryptoProvider implements cryptographic primitives (algorithms) supported by the stack. Implementation of this component may be software or hardware based (HSM/TPM). **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cryptoProvider Documentation	Documentation	0..1	ref	Documentation of the CryptoProvider that describes the implemented cryptographic primitives. **Tags:**atp.Status=draft
keySlot	CryptoKeySlot	*	aggr	This aggregation represents the key slots that are allocated by the CryptoProvider. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=keySlot.shortName atp.Status=draft

Table 9.70: CryptoProvider

Please note that each CryptoProvider can be described in more detail with the cryptoProviderDocumentation that uses the Documentation element to provide means for a more detailed description.

[TPS_MANI_03262]{DRAFT} **Semantics of CryptoProviderToPortPrototypeMapping** ⌈Meta-class CryptoProviderToPortPrototypeMapping has the ability to map a specific PortPrototype referenced in the role portPrototype to a CryptoProvider referenced in the role cryptoProvider.

The mapping also comprises a reference to meta-class process in order to accommodate for the fact that identical combinations of cryptoProvider and portPrototype may or may not apply for a given Process that represents the enclosing Executable at runtime.⌋*(RS_MANI_00023)*

[constr_5240]{DRAFT} **Restriction applicable for CryptoProviderToPortPrototypeMapping.portPrototype** ⌈The reference CryptoProviderToPortPrototypeMapping.portPrototype shall only be used for an RPortPrototype typed by a CryptoProviderInterface.⌋*()*

The application developer uses the ara::core::InstanceSpecifier as local identifier in the API call that represents the path to the modeled RPortPrototype. The Integrator maps the RPortPrototype in the deployment model with the CryptoProviderToPortPrototypeMapping to the concrete CryptoProvider.

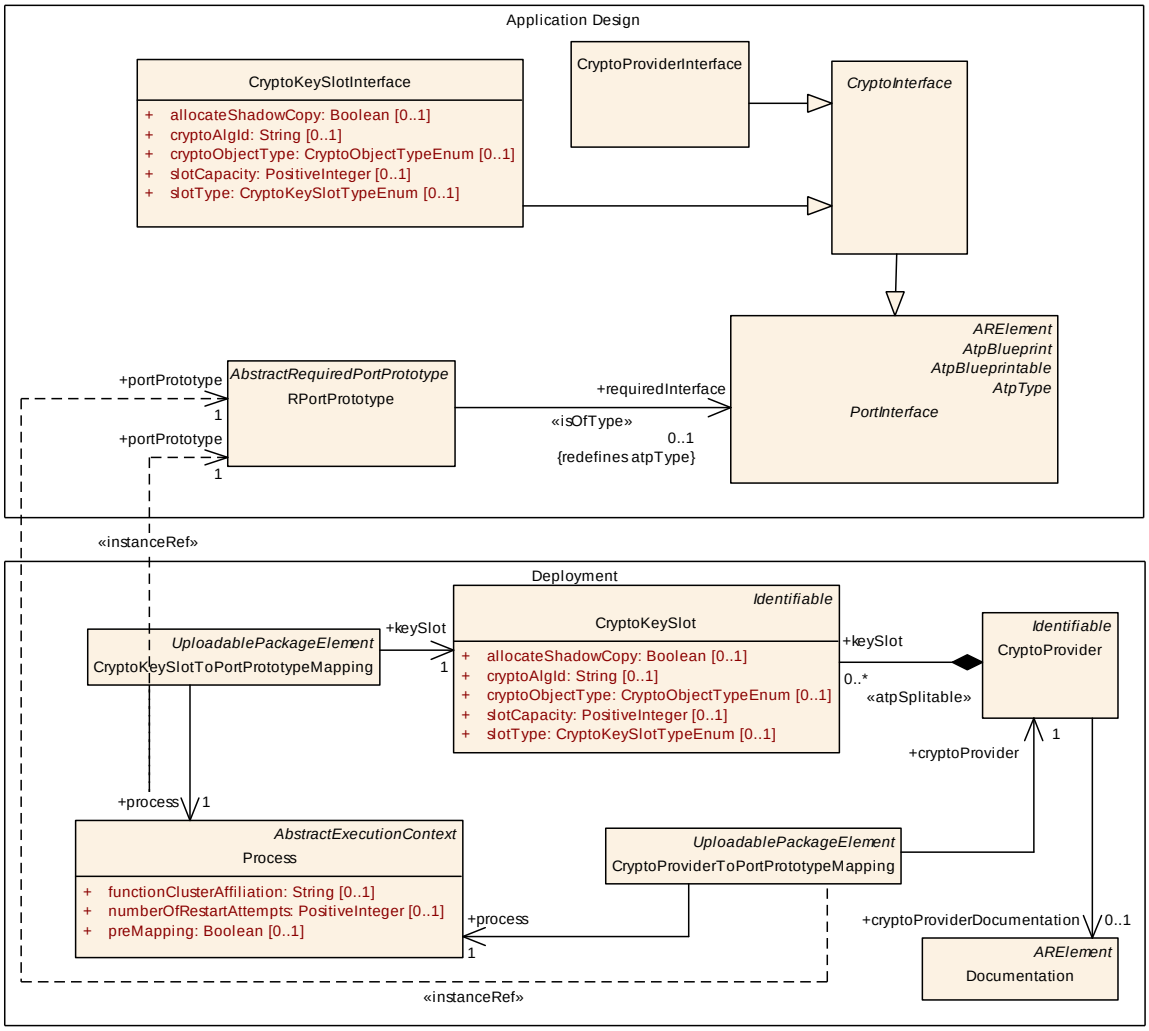


Figure 9.34: Mapping of CryptoProviders and CryptoKeySlots to RPortPrototypes

9.10.2 Crypto Key Slot

The Key Storage Provider is responsible for the storage of different type of key material. The crypto objects that are stored by the Key Storage Provider are represented as `CryptoKeySlot`s in the model.

[TPS_MANI_03263]{DRAFT} **Assignment of `CryptoKeySlot`s to `CryptoProvider`s** ⌈Crypto objects that are used by the `CryptoProvider` are described by the `CryptoKeySlot`s that are aggregated by the `CryptoProvider` in the role `keySlot`.⌋*(RS_MANI_00023)*

Class	**CryptoKeySlot**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to define a concrete key to be used for a crypto operation. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allocateShadow Copy	Boolean	0..1	attr	This attribute defines whether a shadow copy of this Key Slot shall be allocated to enable rollback of a failed Key Slot update campaign (see interface BeginTransaction).
cryptoAlgId	String	0..1	attr	This attribute defines a crypto algorithm restriction (kAlgId Any means without restriction). The algorithm can be specified partially: family & length, mode, padding. Future Crypto Providers can support some crypto algorithms that are not well known/ standardized today, therefore AUTOSAR doesn't provide a concrete list of crypto algorithms' identifiers and doesn't suppose usage of numerical identifiers. Instead of this a provider supplier should provide string names of supported algorithms in accompanying documentation. The name of a crypto algorithm shall follow the rules defined in the specification of cryptography for Adaptive Platform.
cryptoObject Type	CryptoObjectTypeEnum	0..1	attr	Object type that can be stored in the slot. If this field contains "Undefined" then mSlotCapacity must be provided and larger then 0.
keySlotAllowed Modification	CryptoKeySlotAllowed Modification	0..1	aggr	Restricts how this keySlot may be used **Tags:**atp.Status=draft
keySlotContent AllowedUsage	CryptoKeySlotContent AllowedUsage	*	aggr	Restriction of allowed usage of a key stored to the slot. **Tags:**atp.Status=draft
slotCapacity	PositiveInteger	0..1	attr	Capacity of the slot in bytes to be reserved by the stack vendor. One use case is to define this value in case that the cryptoObjectType is undefined and the slot size can not be deduced from cryptoObjectType and cryptoAlgId. "0" means slot size can be deduced from cryptoObject Type and cryptoAlgId.
slotType	CryptoKeySlotType Enum	0..1	attr	This attribute defines whether the keySlot is exclusively used by the Application; or whether it is used by Stack Services and managed by a Key Manager Application.

Table 9.71: CryptoKeySlot

[TPS_MANI_03264]{DRAFT} **Semantics of `CryptoKeySlotToPortPrototypeMapping`** ⌈Meta-class `CryptoKeySlotToPortPrototypeMapping` has the ability to map a specific `PortPrototype` referenced in the role `portPrototype` to a `CryptoKeySlot` referenced in the role `keySlot`.

The mapping also comprises a reference to meta-class `process` in order to accommodate for the fact that identical combinations of `keySlot` and `portPrototype` may or may not apply for a given `Process` that represents the enclosing `Executable` at runtime.⌋*(RS_MANI_00023)*

[constr_5241]{DRAFT} **Restriction applicable for `CryptoKeySlotToPortPrototypeMapping.portPrototype`** ⌈The reference `CryptoKeySlotToPortPrototypeMapping.portPrototype` shall only be used for an `RPortPrototype` typed by a `CryptoKeySlotInterface`.⌋*()*

The application developer uses the `ara::core::InstanceSpecifier` as local identifier in the API call that represents the path to the modeled `RPortPrototype`. The Integrator maps the `RPortPrototype` in the deployment model with the `CryptoKeySlotToPortPrototypeMapping` to the concrete `CryptoKeySlot` that is stored by the Key Storage Provider. The information from the deployment model is therefore used to replace the local identifier from the Application Design by the concrete `CryptoKeySlot`.

9.10.3 Crypto Certificate

[TPS_MANI_03265]{DRAFT} **Support of `CryptoCertificate`s** ⌈Certificates stored by the Certificate Management Provider that is available in the Crypto functional cluster are modeled as `CryptoCertificate`s.⌋*(RS_MANI_00023)*

Class	**CryptoCertificate**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to model a cryptographic certificate. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
isPrivate	Boolean	0..1	attr	This attribute controls the possibility to access the content of the CryptoCertificateSlot by Find() interfaces of the X509 Provider.

Table 9.72: CryptoCertificate

[TPS_MANI_03266]{DRAFT} **Semantics of `CryptoCertificateToCryptoKeySlotMapping`** ⌈The `CryptoCertificateToCryptoKeySlotMapping` is used to assign a private key and optionally a public key to the `CryptoCertificate`.⌋ *(RS_MANI_00023)*

Class	**CryptoCertificateToCryptoKeySlotMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to define a mapping between a CryptoKeySlot and a Crypto Certificate. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
crypto Certificate	CryptoCertificate	1	ref	This reference represents the mapped cryptoCertificate. **Tags:**atp.Status=draft

▽

△

Class	**CryptoCertificateToCryptoKeySlotMapping**			
cryptoKeySlot	CryptoKeySlot	0..2	ref	This reference represents the mapped cryptoKeySlot. **Tags:**atp.Status=draft

Table 9.73: CryptoCertificateToCryptoKeySlotMapping

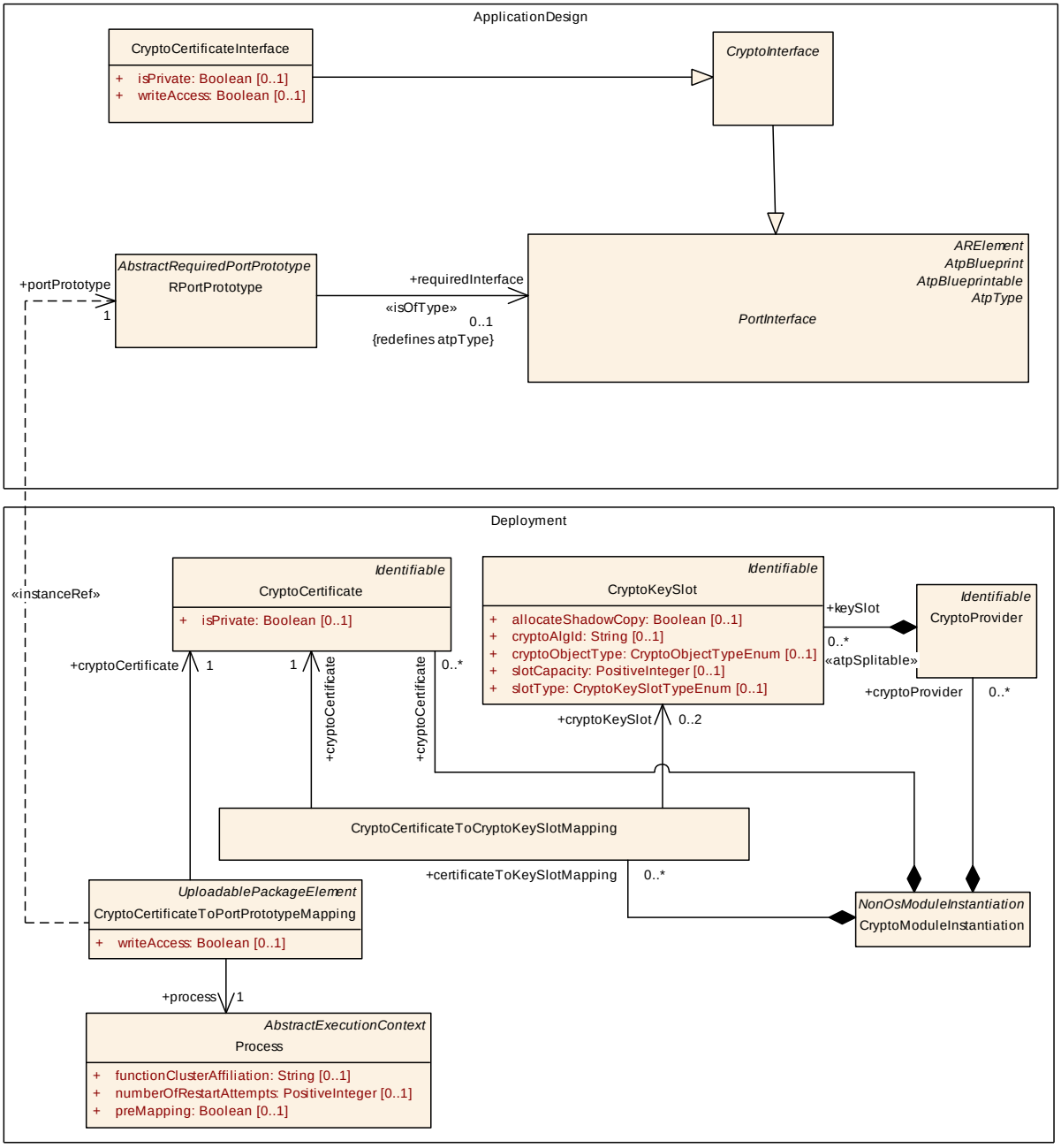


Figure 9.35: Mapping of CryptoCertificates to RPortPrototypes

[TPS_MANI_03267]{DRAFT} **Semantics of `CryptoCertificateToPortPrototypeMapping`** ⌈Meta-class `CryptoCertificateToPortPrototypeMapping` has the ability to map a specific `PortPrototype` referenced in the role `portPrototype` to a `CryptoCertificate` referenced in the role `cryptoCertificate`.

The mapping also comprises a reference to meta-class process in order to accommodate for the fact that identical combinations of cryptoCertificate and portPrototype may or may not apply for a given Process that represents the enclosing Executable at runtime.⌋(*RS_MANI_00023*)

[constr_5242]{DRAFT} **Restriction applicable for CryptoCertificateToPortPrototypeMapping.portPrototype** ⌈The reference CryptoCertificateToPortPrototypeMapping.portPrototype shall only be used for an RPortPrototype typed by a CryptoCertificateInterface.⌋()

The application developer uses the ara::core::InstanceSpecifier as local identifier in the API call that represents the path to the modeled RPortPrototype. The Integrator maps the RPortPrototype in the deployment model with the CryptoCertificateToPortPrototypeMapping to the concrete CryptoCertificate that is stored by the Certificate Management Provider. The information from the deployment model is therefore used to replace the local identifier from the Application Design by the concrete CryptoCertificate.

9.11 IdsM Deployment

9.11.1 IdsM Instantiation

The definition of the deployment for the Intrusion Detection System Manager (IdsM) is modeled by means of the meta-class IdsmModuleInstantiation

Class	**IdsmModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IntrusionDetectionSystem			
Note	This meta-class defines the attributes for the IdsM configuration on a specific machine. **Tags:**atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *IdsPlatformInstantiation*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
reportable SecurityEvent	SecurityEventMapping	*	ref	Collection of reportable instances of security events. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=reportableSecurityEvent atp.Status=draft

Table 9.74: IdsmModuleInstantiation

[constr_10021]{DRAFT} **Existence of IdsmModuleInstantiation** ⌈On each Machine, only one instance of the Intrusion Detection System Manager (modeled by IdsmModuleInstantiation) shall exist.⌋()

This instance manages all the reported SEvs created by SWCLs or Function Clusters on this Adaptive Machine.

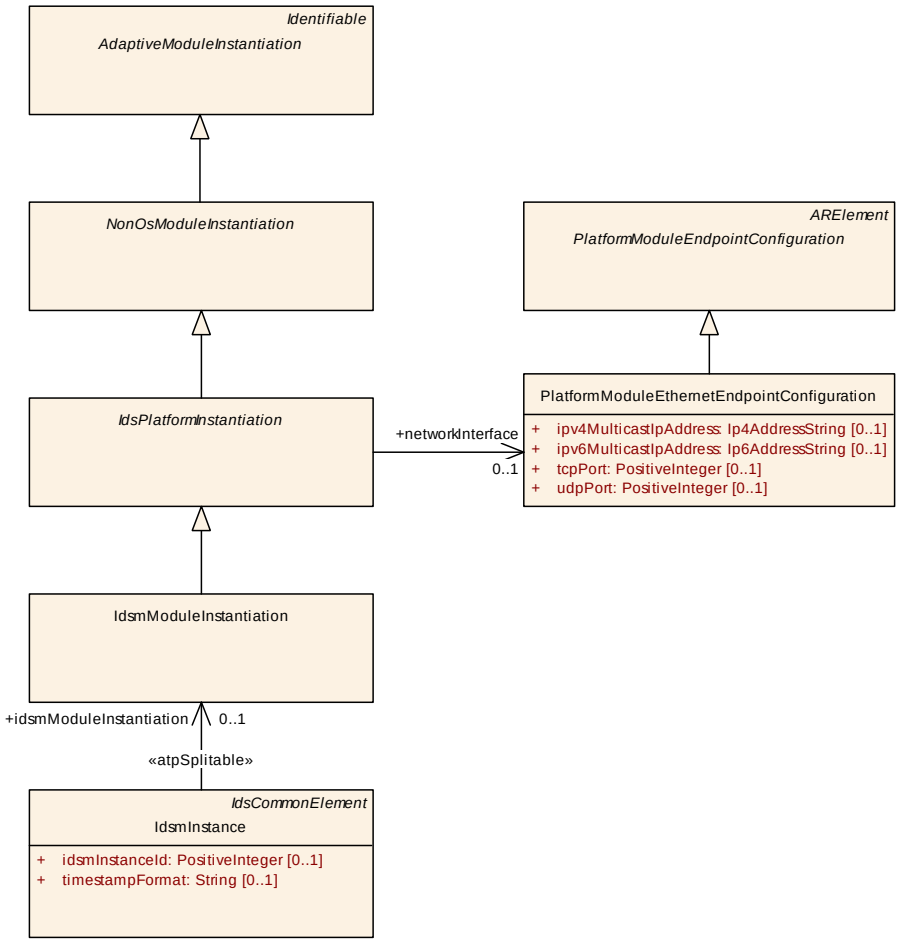


Figure 9.36: Specification of the `IdsmModuleInstantiation`

Meta-class `IdsmModuleInstantiation` is derived from the abstract meta-class `IdsPlatformInstantiation` which acts as an abstract base class for all platform modules that contribute to the implementation of the Intrusion Detection System.

Class	***IdsPlatformInstantiation*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IntrusionDetectionSystem			
Note	This meta-class acts as an abstract base class for platform modules that implement the intrusion detection system. **Tags:**atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Subclasses	IdsmModuleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
network Interface	PlatformModuleEthernetEndpointConfiguration	0..1	ref	This association contains the network configuration that shall be applied to an instance of an IDS entity. **Tags:**atp.Status=draft

△

Class	***IdsPlatformInstantiation*** (abstract)			
timeBase	TimeBaseResource	0..1	ref	This reference identifies the applicable time base resource. **Stereotypes:** atpVariation **Tags:** atp.Status=draft vh.latestBindingTime=systemDesignTime

Table 9.75: IdsPlatformInstantiation

9.11.2 Obtaining custom Time Stamps for Security Events

One of the tasks of an IdsM is to obtain a custom time stamp that corresponds to reported security events under certain circumstances (which are not relevant for the sake of discussing the modeling).

Time stamps can be obtained from sub-classes of the abstract base-class `TimeBaseResource`. Therefore, `IdsmModuleInstantiation` maintains a reference to `TimeBaseResource`.

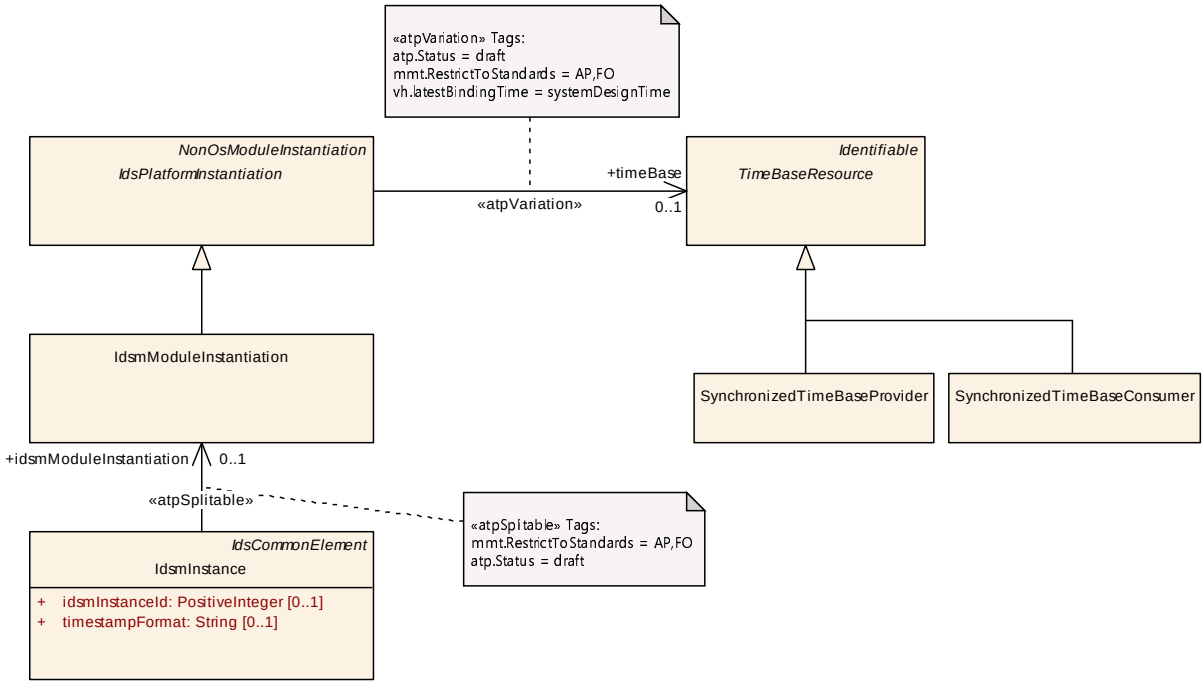


Figure 9.37: Creating an association between a `IdsmModuleInstantiation` and a `TimeBaseResource`

9.11.3 Deployment for Security Events

[TPS_MANI_01341]{DRAFT} **Security events that are actually reported by a local IdsM** ⌈The security events that are actually reported by a local IdsM are represented by

meta-class `SecurityEventMapping`, referenced by `IdsmModuleInstantiation` in the role `reportableSecurityEvent`.⌋()

[TPS_MANI_01342]{DRAFT} **Semantics of `SecurityEventMapping`** ⌈The semantics of meta-class `SecurityEventMapping` is to identify the

- `PortPrototype` in the context of an `Executable` from which the security event is reported
- `Process` that runs the `Executable`, and from the `Process` the `SecurityEventDefinition` from which the `SecurityEventMapping.id` has been derived.

⌋()

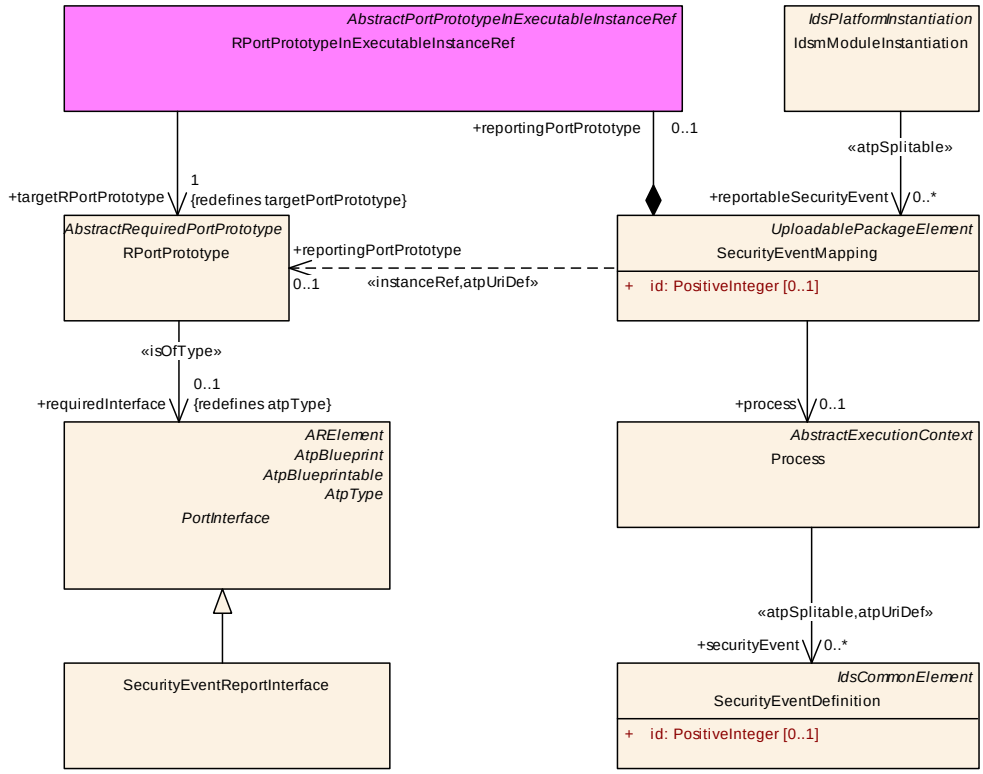


Figure 9.38: Modeling of between a `SecurityEventMapping`

Class	**SecurityEventMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IntrusionDetectionSystem
Note	This meta-class represents a reportable instance of a security event. **Tags:** atp.Status=draft atp.recommendedPackage=SecurityEventMappings
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*

▽

△

Class	**SecurityEventMapping**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
id	PositiveInteger	0..1	attr	This attribute defines the numerical identification of the security event subject to deployment.
process	Process	0..1	ref	This reference identifies the process in which context the seurity event is reported. **Tags:**atp.Status=draft
reportingPort Prototype	RPortPrototype	0..1	iref	This instanceRef identifies the portPrototype over which the security event is reported. **Tags:**atp.Status=draft **InstanceRef implemented by:**RPortPrototypeIn ExecutableInstanceRef

Table 9.76: SecurityEventMapping

[constr_10022]{DRAFT} **Restriction for `SecurityEventMapping.process.securityEvent.id` w.r.t `SecurityEventMapping.id`** ⌈The value of `SecurityEventMapping.id` shall also occur in one of the `SecurityEventDefinition.id` referenced in the role `SecurityEventMapping.process.securityEvent` **at the time when the creation of the manifest is finished**.⌋*()*

Rationale for [constr_10022]: during the creation of the IdsM deployment, the value of `SecurityEventMapping.id` shall be copied from one of the values in `SecurityEventMapping.process.securityEvent.id`.

10 Service Instance Manifest

10.1 Service Interface Deployment

The different meta-class specializations of `ServiceInterfaceDeployment` define a binding of a `ServiceInterface` to a middleware transport layer.

This chapter describes the usage of the `ServiceInterfaceDeployment` in different bindings that are supported by AUTOSAR.

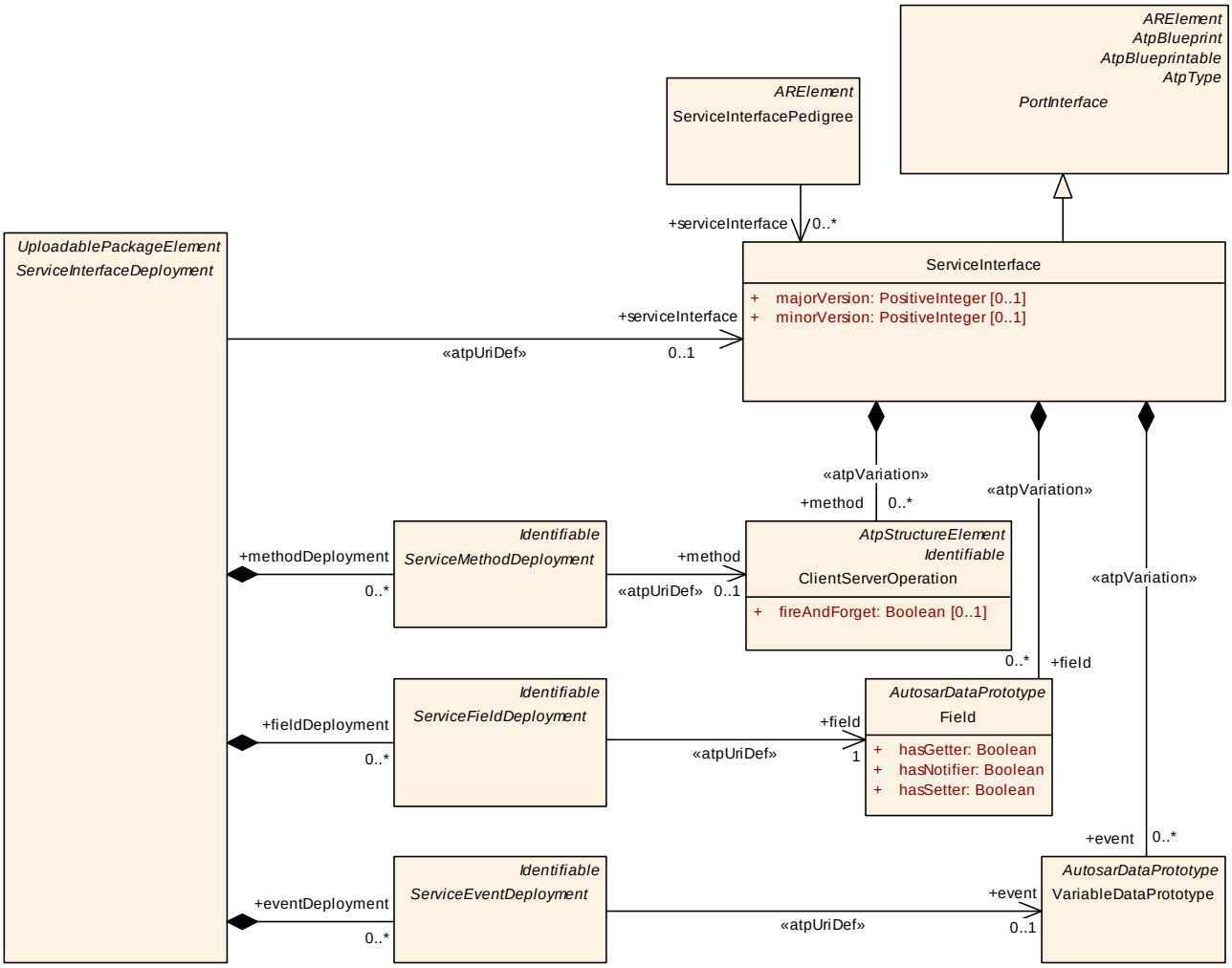


Figure 10.1: Deployment-related modeling of ServiceInterface

[TPS_MANI_03036]{DRAFT} **`ServiceInterface` deployment to a middleware transport layer** ⌈The `ServiceInterfaceDeployment` meta-class provides the ability to map a `ServiceInterface` to a middleware transport layer that is represented by a concrete class that is derived from the abstract `ServiceInterfaceDeployment` meta-class.⌋ *(RS_MANI_00008)*

The association between the ServiceInterfaceDeployment and the ServiceInterface implicitly also defines the relation between the technology specific version number of the service on the ServiceInterfaceDeployment and the service version of the ServiceInterface (defined by ServiceInterface.majorVersion and ServiceInterface.minorVersion)

[TPS_MANI_03617]{DRAFT} **Version mapping between ServiceInterface and ServiceInterfaceDeployment** ⌈The contract version of a ServiceInterface (majorVersion, minorVersion) shall be mapped to a version of the ServiceInterfaceDeployment for each transport layer.

This version mapping may lead to different version numbers for different ServiceInterfaceDeployments that refer to the same ServiceInterface. This allows to define different version numbers, on the same network or on different networks (e.g. VLANs).⌋*(RS_MANI_00065)*

Note that transport layer specific constraints on the uniqueness of protocol credentials still have to be respected, e.g. [constr_1723].

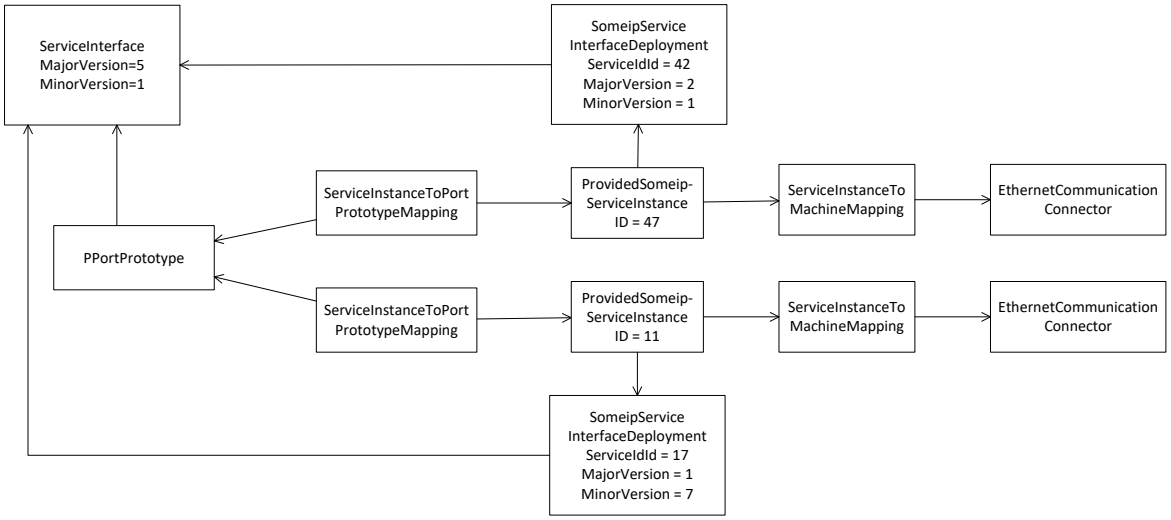


Figure 10.2: Example of 1 PPortPrototype mapped to 2 VLANs

In figure 10.2 the use-case of having one PPortPrototype typed by one ServiceInterface and having several ProvidedSomeipServiceInstances on several VLANs is illustrated. It is possible to have different serviceInterfaceId, serviceInstanceId, and majorVersion values on each individual VLAN. But also the use-case of providing both service instances on the same VLAN would be supported, as long as their SOME/IP credentials are unique (see [constr_1723]).

Class	***ServiceInterfaceDeployment*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	Middleware transport layer specific configuration settings for the ServiceInterface and all contained ServiceInterface elements. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Subclasses	DdsServiceInterfaceDeployment, SomeipServiceInterfaceDeployment, UserDefinedServiceInterface Deployment			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
event Deployment	ServiceEvent Deployment	*	aggr	Middleware transport layer specific configuration settings for an Event that is defined in the ServiceInterface. **Tags:**atp.Status=draft
fieldDeployment	ServiceField Deployment	*	aggr	Middleware transport layer specific configuration settings for a Field that is defined in the ServiceInterface. **Tags:**atp.Status=draft
method Deployment	ServiceMethod Deployment	*	aggr	Middleware transport layer specific configuration settings for a method that is defined in the ServiceInterface. **Tags:**atp.Status=draft
serviceInterface	ServiceInterface	0..1	ref	Reference to a ServiceInterface that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:**atp.Status=draft

Table 10.1: ServiceInterfaceDeployment

[TPS_MANI_03037]{DRAFT} **Purpose of `ServiceMethodDeployment`** ⌈The `ServiceMethodDeployment` meta-class provides the ability to define middleware transport layer specific configuration settings relevant for a `method` that is defined in the context of a `ServiceInterface`.⌋*(RS_MANI_00008)*

[constr_3300]{DRAFT} **Allowed `ServiceMethodDeployment.method` references** ⌈The `ClientServerOperation` that is referenced by `ServiceMethodDeployment` in the role `method` shall be defined in the context of a `ServiceInterface` that is referenced by the `ServiceInterfaceDeployment` in the role `serviceInterface` that contains the `ServiceMethodDeployment`.⌋*()*

[TPS_MANI_03038]{DRAFT} **Purpose of `ServiceEventDeployment`** ⌈The `ServiceEventDeployment` meta-class provides the ability to define middleware transport layer specific configuration settings relevant for an `event` that is defined in the context of a `ServiceInterface`.⌋*(RS_MANI_00008)*

[constr_3301]{DRAFT} **Allowed `ServiceEventDeployment.event` references** ⌈The `VariableDataPrototype` that is referenced by `ServiceEventDeployment` in the role `event` shall be defined in the context of a `ServiceInterface` that is referenced by the `ServiceInterfaceDeployment` in the role `serviceInterface` that contains the `ServiceEventDeployment`.⌋*()*

[TPS_MANI_03039]{DRAFT} **Purpose of `ServiceFieldDeployment`** ⌈The `ServiceFieldDeployment` meta-class provides the ability to define middleware transport layer specific configuration settings relevant for a `field` that is defined in the context of a `ServiceInterface`.⌋ *(RS_MANI_00008)*

[constr_3302]{DRAFT} **Allowed `ServiceFieldDeployment.field` references** ⌈The `Field` that is referenced by `ServiceFieldDeployment` in the role `field` shall be defined in the context of a `ServiceInterface` that is referenced by the `ServiceInterfaceDeployment` in the role `serviceInterface` that contains the `ServiceFieldDeployment`.⌋ *()*

Class	***ServiceMethodDeployment*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	This abstract meta-class represents the ability to specify a deployment of a Method to a middleware transport layer. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	SomeipMethodDeployment, UserDefinedMethodDeployment			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
method	ClientServerOperation	0..1	ref	Reference to a method that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:**atp.Status=draft

Table 10.2: ServiceMethodDeployment

Class	***ServiceEventDeployment*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	This abstract meta-class represents the ability to specify a deployment of an Event to a middleware transport layer. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	DdsEventDeployment, SomeipEventDeployment, UserDefinedEventDeployment			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
event	VariableDataPrototype	0..1	ref	Reference to an Event that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:**atp.Status=draft

Table 10.3: ServiceEventDeployment

Class	***ServiceFieldDeployment*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment
Note	This abstract meta-class represents the ability to specify a deployment of a Field to a middleware transport layer. **Tags:**atp.Status=draft
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	***ServiceFieldDeployment*** (abstract)			
Subclasses	DdsFieldDeployment, SomeipFieldDeployment, UserDefinedFieldDeployment			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
field	Field	1	ref	Reference to a Field that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:**atp.Status=draft

Table 10.4: ServiceFieldDeployment

10.1.1 SOME/IP Service Interface Deployment

This chapter describes the SOME/IP deployment of a `ServiceInterface`.

[TPS_MANI_03040]{DRAFT} **SOME/IP ServiceInterface binding** ⌈The `SomeipServiceInterfaceDeployment` meta-class provides the ability to bind a `ServiceInterface` to SOME/IP and to assign a SOME/IP Service identifier to the `ServiceInterface` with the `serviceInterfaceId` attribute.⌋*(RS_MANI_00024)*

The idea behind the `SomeipServiceInterfaceDeployment` is the definition of a common configuration set that is shared between the server that provides the `ServiceInterface` and all clients that are consuming the `ServiceInterface`. So it contains all relevant SOME/IP settings used for identification of the `ServiceInterface` and its content in messages on the network.

[constr_3410]{DRAFT} **Value range of `SomeipServiceInterfaceDeployment.serviceInterfaceId`** ⌈The value of `serviceInterfaceId` shall be in the range of 0..65535.⌋*()*

Please note that the SOME/IP MessageId that is 32 Bit long contains a 16 Bit `serviceInterfaceId`, a single bit that defines whether the message transports a method or an event and a 15 Bit `eventId` or `methodId`.

Please also consider [PRS_SOMEIPSD_00515] in [24] that defines special and reserved `serviceInterfaceId`s for SOME/IP and SOME/IP-SD.

[TPS_MANI_03041]{DRAFT} **Definition of SOME/IP EventGroups** ⌈The `SomeipServiceInterfaceDeployment.eventGroup` allows to define SOME/IP *EventGroups* that are included in the SOME/IP Service and provide a logical grouping of events and notification events used for publish/subscribe handling.⌋*(RS_MANI_00024)*

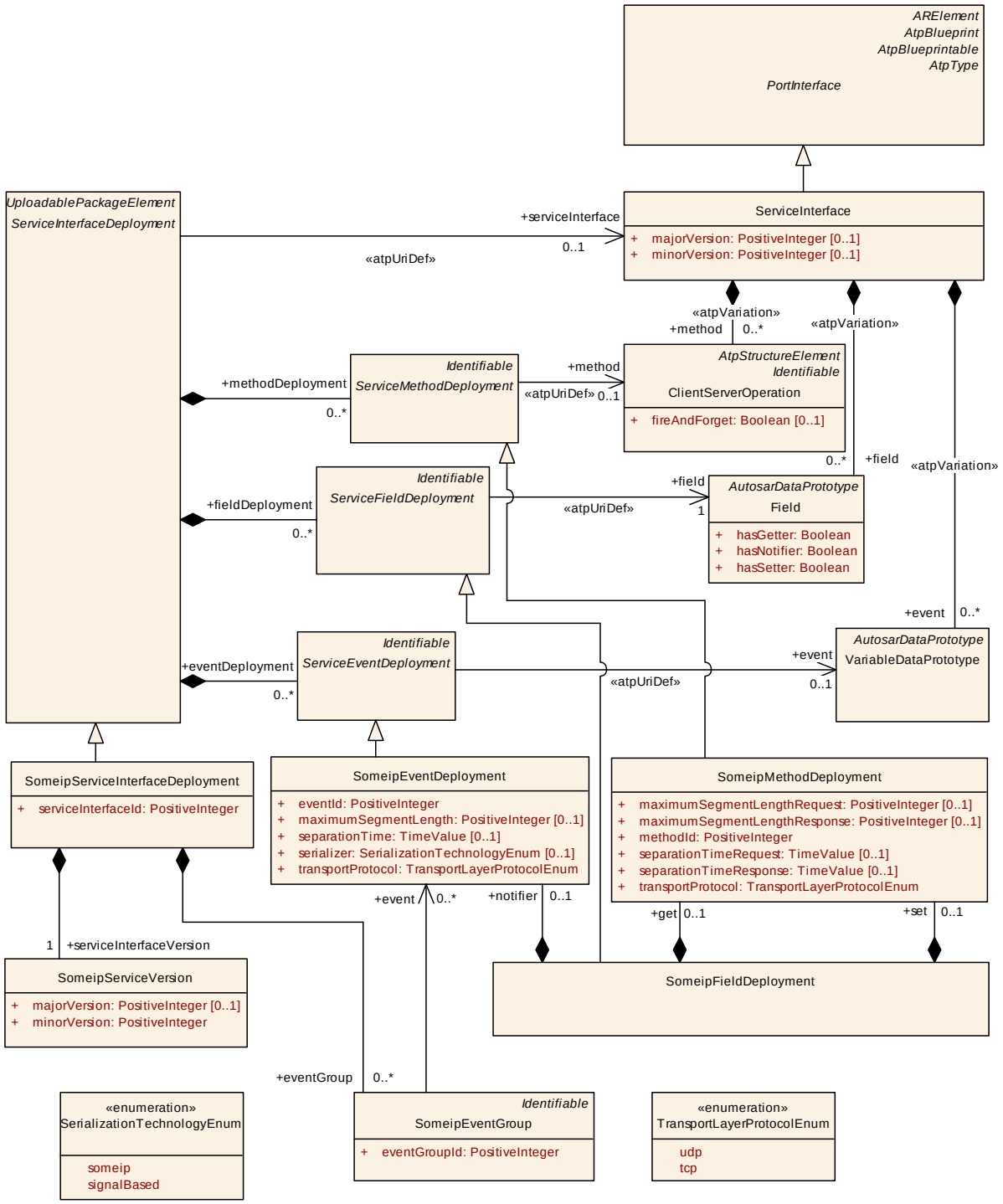


Figure 10.3: SOME/IP deployment of ServiceInterface

[constr_3304]{DRAFT} **Value of attribute `SomeipEventGroup.eventGroupId` shall be unique** ⌈The value of attribute `eventGroupId` shall be unique in the context of the enclosing `SomeipServiceInterfaceDeployment`.⌋*()*

[TPS_MANI_03042]{DRAFT} **Definition of SOME/IP Service Version** ⌈The `SomeipServiceInterfaceDeployment.serviceInterfaceVersion` allows to define a major and a minor version for the SOME/IP Service.⌋*(RS_MANI_00024)*

[constr_3557]{DRAFT} **Mandatory `majorVersion` at `SomeipServiceInterfaceDeployment.serviceInterfaceVersion`** ⌈If the `SomeipServiceVersion` is aggregated at the `SomeipServiceInterfaceDeployment` in the role `serviceInterfaceVersion` then the attribute `SomeipServiceVersion.majorVersion` shall be defined.⌋*()*

Class	**SomeipServiceInterfaceDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for a ServiceInterface. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInterfaceDeployments			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *ServiceInterfaceDeployment*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventGroup	SomeipEventGroup	*	aggr	SOME/IP EventGroups that are defined within the SOME/IP ServiceClass. **Tags:**atp.Status=draft
serviceInterfaceId	PositiveInteger	1	attr	Unique Identifier that identifies the ServiceInterface in SOME/IP. This Identifier is sent as Service ID in SOME/IP Service Discovery messages.
serviceInterfaceVersion	SomeipServiceVersion	1	aggr	The SOME/IP major and minor Version of the Service. **Tags:**atp.Status=draft

Table 10.5: SomeipServiceInterfaceDeployment

Class	**SomeipEventGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	Grouping of events and notification events inside a ServiceInterface in order to allow subscriptions. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
event	SomeipEventDeployment	*	ref	Reference to an event that is part of the EventGroup. **Tags:**atp.Status=draft
eventGroupId	PositiveInteger	1	attr	Unique Identifier that identifies the EventGroup in SOME/IP. This Identifier is sent as Eventgroup ID in SOME/IP Service Discovery messages.

Table 10.6: SomeipEventGroup

[TPS_MANI_03043]{DRAFT} **SOME/IP `VariableDataPrototype` binding** ⌈The `SomeipEventDeployment` meta-class provides the ability to bind a `VariableDataPrototype` to SOME/IP and to assign a SOME/IP Event identifier to the `event` with the `eventId` attribute.⌋*(RS_MANI_00024)*

[constr_3305]{DRAFT} **Value of attribute `SomeipEventDeployment.eventId` shall be unique** ⌈The value of `eventId` shall be unique in the context of the enclosing `SomeipServiceInterfaceDeployment`.⌋*()*

[constr_3408]{DRAFT} **Value range of `SomeipEventDeployment.eventId`** ⌈The value of `eventId` shall be in the range of 0..32767.⌋*()*

Please note that [PRS_SOMEIPSD_00517] in [24] defines special and reserved EVENT-IDs for SOME/IP and SOME/IP-SD that result in the eventId values of 0 and 32767.

[TPS_MANI_03050]{DRAFT} **Usage of SomeipEventDeployment.transportProtocol** ⌈The value of SomeipEventDeployment.transportProtocol defines over which Transport Layer Protocol the SomeipEventDeployment.event is provided.⌋*(RS_MANI_00024)*

[constr_5156]{DRAFT} **SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances** ⌈If SomeipEventDeployment.transportProtocol is set to udp then each ProvidedSomeipServiceInstance that refers the SomeipServiceInterfaceDeployment in the role serviceInterfaceDeployment shall only be mapped to a MachineDesign with a SomeipServiceInstanceToMachineMapping with a configured udpPort.⌋*()*

[constr_3308]{DRAFT} **SomeipEventDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances** ⌈If SomeipEventDeployment.transportProtocol is set to tcp then each ProvidedSomeipServiceInstance that refers the SomeipServiceInterfaceDeployment in the role serviceInterfaceDeployment shall only be mapped to a MachineDesign with a SomeipServiceInstanceToMachineMapping with a configured tcpPort.⌋*()*

[TPS_MANI_03067]{DRAFT} **SOME/IP segmentation of udp SomeipEventDeployments** ⌈If the maximumSegmentLength is set to a value and the data length is larger than maximumSegmentLength then SOME/IP shall segment the SomeipEventDeployment into several packets and transmit them over the network.

The sender shall wait the separationTime between the transmissions of segments. On the reception side, SOME/IP re-assembles the received SOME/IP segments to the original SomeipEventDeployment.⌋*(RS_MANI_00024)*

[constr_3351]{DRAFT} **SOME/IP segmentation allowed for udp SomeipEventDeployments** ⌈Attribute SomeipEventDeployment.maximumSegmentLength shall only be used if the value of attribute SomeipEventDeployment.transportProtocol is set to udp.⌋*()*

As the SomeipServiceInterfaceDeployment is also used for the deployment of signal-based Pdus on Ethernet the attribute SomeipEventDeployment.serializer defines whether the someip or the signalBased serialization shall be used for a specific event.

[TPS_MANI_03615]{DRAFT} **SomeipEventDeployment.serializer equals someip** ⌈If the attribute SomeipEventDeployment.serializer is not defined or is set to the value someip then the event shall be serialized/de-serialized using the SOME/IP serializer.⌋*(RS_MANI_00063)*

[TPS_MANI_03591]{DRAFT} **SomeipEventDeployment.serializer equals signalBased** ⌈If the attribute SomeipEventDeployment.serializer is set to the value signalBased then the event shall be serialized/de-serialized using the signal-based approach.⌋ *(RS_MANI_00063)*

This aspect is described in chapter 11.

Class	**SomeipEventDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for an Event. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceEventDeployment*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventId	PositiveInteger	1	attr	Unique Identifier within a ServiceInterface that identifies the Event in SOME/IP. This Identifier is sent as part of the Message ID in SOME/IP messages.
maximum SegmentLength	PositiveInteger	0..1	attr	This attribute describes the length in bytes of the SOME/IP segment. This includes 8 bytes for the Request ID, Protocol Version, Interface Version, Message Type and Return Code and 4 additional SOME/IP TP bytes. If this attribute is set to a value and the data length is larger than maximumSegmentLength then the corresponding SOME/IP message will be segmented into smaller parts that are transmitted over the network.
separationTime	TimeValue	0..1	attr	Sets the duration of the minimum time in seconds SOME/IP shall wait between the transmissions of segments.
serializer	SerializationTechnology Enum	0..1	attr	Defines which serialization technology shall be used.
transport Protocol	TransportLayerProtocol Enum	1	attr	This attribute defines over which Transport Layer Protocol this event is intended to be sent.

Table 10.7: SomeipEventDeployment

Enumeration	**SerializationTechnologyEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment
Note	This enumeration allows to choose a Serialization Technology. **Tags:**atp.Status=draft
Literal	***Description***
signalBased	Signal-Based serializer. **Tags:**atp.EnumerationLiteralIndex=1
someip	SOME/IP Serializer **Tags:**atp.EnumerationLiteralIndex=0

Table 10.8: SerializationTechnologyEnum

[TPS_MANI_03044]{DRAFT} **SOME/IP ClientServerOperation binding** ⌈The SomeipMethodDeployment meta-class provides the ability to bind a ClientServerOperation to SOME/IP and to assign a SOME/IP Method identifier to the method with the methodId attribute.⌋ *(RS_MANI_00024)*

[constr_3306]{DRAFT} **Value of attribute methodId shall be unique per SomeipServiceInterfaceDeployment** ⌈The value of methodId shall be unique in the context of the enclosing SomeipServiceInterfaceDeployment.⌋ *()*

[constr_3409]{DRAFT} **Value range of SomeipMethodDeployment.methodId** ⌈The value of methodId shall be in the range of 0..32767.⌋*()*

Please note that [PRS_SOMEIPSD_00517] in [24] defines special and reserved METHOD-IDs for SOME/IP and SOME/IP-SD that result in the methodId values of 0 and 32767.

[TPS_MANI_03051]{DRAFT} **Usage of SomeipMethodDeployment.transportProtocol** ⌈The value of SomeipMethodDeployment.transportProtocol defines over which Transport Layer Protocol this method is provided.⌋*(RS_MANI_00024)*

[constr_3309]{DRAFT} **SomeipMethodDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances** ⌈If SomeipMethodDeployment.transportProtocol is set to udp then each ProvidedSomeipServiceInstance that refers the SomeipServiceInterfaceDeployment in the role serviceInterfaceDeployment shall only be mapped to a MachineDesign with a SomeipServiceInstanceToMachineMapping with a configured udpPort.⌋*()*

[constr_3310]{DRAFT} **SomeipMethodDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances** ⌈If SomeipMethodDeployment.transportProtocol is set to tcp then each ProvidedSomeipServiceInstance that refers the SomeipServiceInterfaceDeployment in the role serviceInterfaceDeployment shall only be mapped to a MachineDesign with a SomeipServiceInstanceToMachineMapping with a configured tcpPort.⌋*()*

[TPS_MANI_03068]{DRAFT} **SOME/IP segmentation of SomeipMethodDeployment Calls** ⌈If the maximumSegmentLengthRequest is set to a value and the data length is larger than maximumSegmentLengthRequest then SOME/IP shall segment the SomeipMethodDeployment Call-Message into several packets and transmit them over the network.

The sender shall wait the separationTimeRequest between the transmissions of segments. On the reception side, SOME/IP re-assembles the received SOME/IP segments to the original SomeipMethodDeployment Call-Message.⌋*(RS_MANI_00024)*

[TPS_MANI_03069]{DRAFT} **SOME/IP segmentation of SomeipMethodDeployment Responses** ⌈If the maximumSegmentLengthResponse is set to a value and the data length is larger than maximumSegmentLengthResponse then SOME/IP shall segment the SomeipMethodDeployment Response-Message into several packets and transmit them over the network.

The sender shall wait the separationTimeResponse between the transmissions of segments. On the reception side, SOME/IP re-assembles the received SOME/IP segments to the original SomeipMethodDeployment Response-Message.⌋*(RS_MANI_00024)*

[constr_3352]{DRAFT} **SOME/IP segmentation allowed for udp SomeipMethodDeployments** ⌈SomeipMethodDeployment.maximumSegmentLengthRequest

and SomeipMethodDeployment.maximumSegmentLengthResponse shall only be used if SomeipMethodDeployment.transportProtocol is set to udp.⌋()

Class	SomeipMethodDeployment			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for a Method. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceMethodDeployment*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
maximum SegmentLength Request	PositiveInteger	0..1	attr	This attribute describes the length in bytes of one SOME/IP segment into which the Method Call Message will be divided. This length field includes 8 bytes for the Request ID, Protocol Version, Interface Version, Message Type and Return Code and 4 additional SOME/IP TP bytes. If this attribute is set to a value and the data length is larger than maximumSegmentLengthRequest then the corresponding SOME/IP message will be segmented into smaller parts that are transmitted over the network.
maximum SegmentLength Response	PositiveInteger	0..1	attr	This attribute describes the length in bytes of one SOME/IP segment into which the Method Return Message will be divided. This length field includes 8 bytes for the Request ID, Protocol Version, Interface Version, Message Type and Return Code and 4 additional SOME/IP TP bytes. If this attribute is set to a value and the data length is larger than maximumSegmentLengthResponse then the corresponding SOME/IP message will be segmented into smaller parts that are transmitted over the network.
methodId	PositiveInteger	1	attr	Unique Identifier within a ServiceInterface that identifies the Method in SOME/IP. This Identifier is sent as part of the Message ID in SOME/IP messages.
separationTime Request	TimeValue	0..1	attr	Sets the duration of the minimum time in seconds SOME/IP shall wait between the transmissions of segments into which the Method Call Message will be divided.
separationTime Response	TimeValue	0..1	attr	Sets the duration of the minimum time in seconds SOME/IP shall wait between the transmissions of segments into which the Method Return Message will be divided.
transport Protocol	TransportLayerProtocol Enum	1	attr	This attribute defines over which Transport Layer Protocol this method is intended to be sent.

Table 10.9: SomeipMethodDeployment

Class	SomeipServiceInstanceToMachineMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	This meta-class allows to map SomeipServiceInstances to a CommunicationConnector of a Machine. In this step the network configuration (IP Address, Transport Protocol, Port Number) for the ServiceInstance is defined. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInstanceToMachineMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInstanceToMachineMapping*, *UploadablePackageElement*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	SomeipServiceInstanceToMachineMapping			
tcpPort	PositiveInteger	0..1	attr	TcpPort configuration that is used for Method and Event communication in IP-Unicast case. During SOME/IP Service Discovery: PortNumber that is sent in the SD-Offer Message to client (answer on SD-find) or clients (SD-offer). Method: This is the destination-port where the server accepts the method call messages (from the clients). This is the source-port where the server sends the method response messages (to the client). Event: This is the event source-port where the server sends the event messages to the subscribed clients in IP-Unicast case.
udpCollection BufferSize Threshold	PositiveInteger	0..1	attr	Specifies the amount of data in bytes that shall be buffered for data transmission over the udp connection specified by this SomeipServiceInstanceToMachine Mapping in case data collection is enabled.
udpPort	PositiveInteger	0..1	attr	UdpPort configuration that is used for Method and Event communication in IP-Unicast case. During SOME/IP Service Discovery: PortNumber that is sent in the SD-Offer Message to client (answer on SD-find) or clients (SD-offer). Method: This is the destination-port where the server accepts the method call messages (from the clients). This is the source-port where the server sends the method response messages (to the client). Event: This is the event source-port where the server sends the event messages to the subscribed clients in IP-Unicast case.

Table 10.10: SomeipServiceInstanceToMachineMapping

[TPS_MANI_03057]{DRAFT} **SOME/IP `Field` binding** ⌈The `SomeipFieldDeployment` meta-class provides the ability to bind a `Field` to SOME/IP.

If the `Field` contains a notifier (`hasNotifier` = true), it is possible to assign a SOME/IP notifier identifier to the `field` by setting the value of attribute `SomeipFieldDeployment.notifier.eventId`.

If the `Field` contains a getter method (`hasGetter` = true), it is possible to assign a SOME/IP notifier identifier to the `field` by setting the value of attribute `SomeipFieldDeployment.get.methodId`.

If the `Field` contains a setter method (`hasSetter` = true), it is possible to assign a SOME/IP notifier identifier to the `field` by setting the value of attribute `SomeipFieldDeployment.set.methodId`⌋ *(RS_MANI_00024)*

Please note that each `methodId` and each `eventId` of a `SomeipFieldDeployment` shall be unique in the context of a `ServiceInterface` as defined in [constr_3306] and [constr_3305].

Class	**SomeipFieldDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for a Field. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceFieldDeployment*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
get	SomeipMethod Deployment	0..1	aggr	This aggregation represents the setting of the get method. **Tags:**atp.Status=draft
notifier	SomeipEvent Deployment	0..1	aggr	This aggregation represents the settings of the notifier. **Tags:**atp.Status=draft
set	SomeipMethod Deployment	0..1	aggr	This aggregation represents the settings of the set method **Tags:**atp.Status=draft

Table 10.11: SomeipFieldDeployment

[constr_3362]{DRAFT} **`SomeipEventDeployment`s aggregated by a `SomeipFieldDeployment`** ⌈A `SomeipEventDeployment` that is aggregated by a `SomeipFieldDeployment` in the role `notifier` shall not reference a `VariableDataPrototype` in the role `event`.⌋*()*

[constr_3363]{DRAFT} **`SomeipMethodDeployment`s aggregated by a `SomeipFieldDeployment`** ⌈A `SomeipMethodDeployment` that is aggregated by a `SomeipFieldDeployment` in the role `get` or `set` shall not reference a `ClientServerOperation` in the role `method`.⌋*()*

[TPS_MANI_03227]{DRAFT} **Usage of ephemeral ports** ⌈Ephemeral ports are short-lived transport protocol ports that are allocated automatically by the communication middleware. In case the port number (`SomeipServiceInstanceToMachineMapping.udpPort` or `SomeipServiceInstanceToMachineMapping.tcpPort`) is configured to 0, an *ephemeral* port shall be used. If the port number is configured to a value different from 0, exactly that value shall be used.⌋*(RS_MANI_00024)*

10.1.2 DDS Service Interface Deployment

This chapter describes the DDS [25] deployment of a `ServiceInterface`.

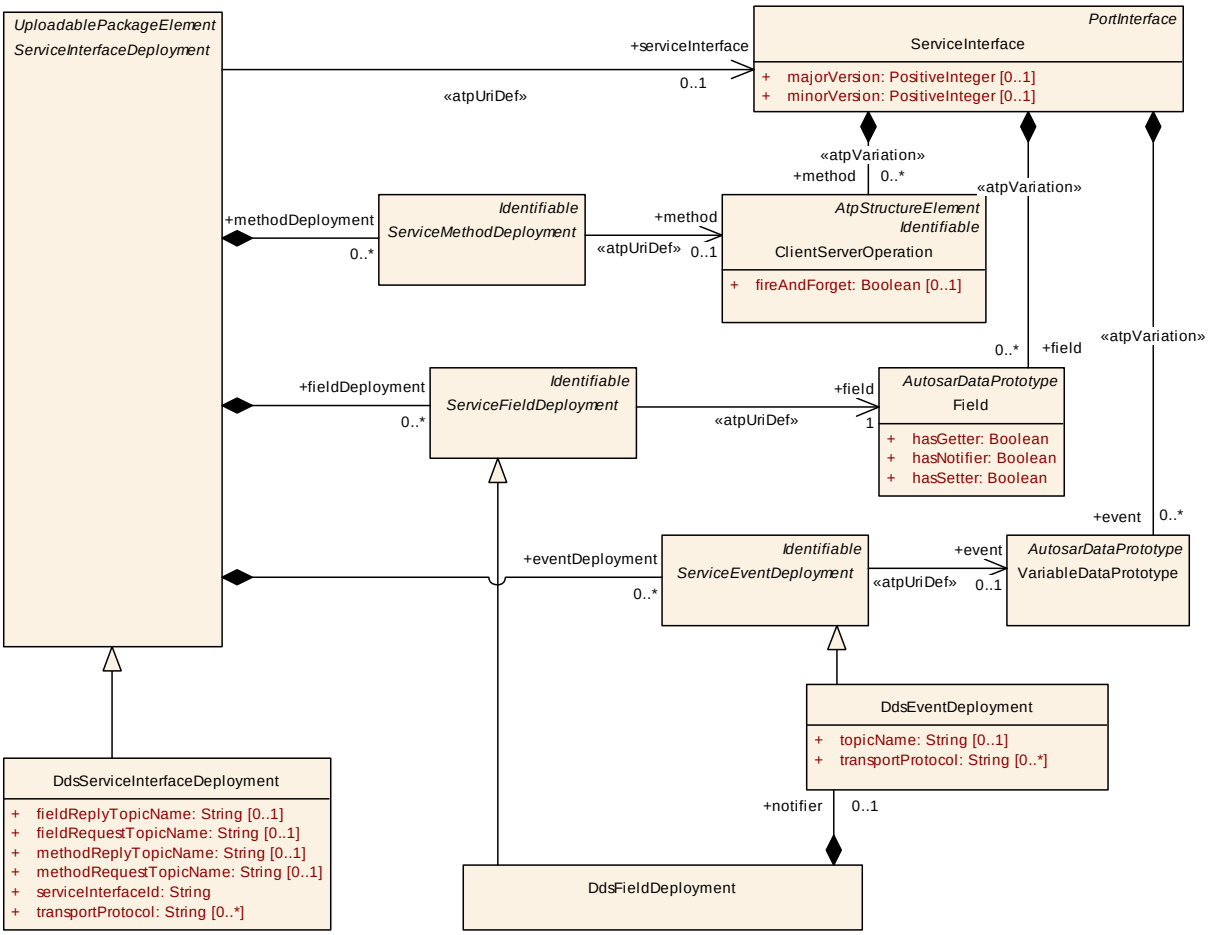


Figure 10.4: DDS deployment of ServiceInterface

[TPS_MANI_03525]{DRAFT} **DDS ServiceInterface binding** ⌈The `DdsServiceInterfaceDeployment` meta-class provides the ability to bind a `ServiceInterface` to DDS and to assign a DDS Service identifier to the `ServiceInterface` with the `serviceInterfaceId` attribute.⌋ *(RS_MANI_00038)*

Class	**DdsServiceInterfaceDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	DDS configuration settings for a ServiceInterface. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInterfaceDeployments			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInterfaceDeployment*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
fieldReplyTopic Name	String	0..1	attr	Name of the DDS Reply Topic associated with the Field.
fieldRequest TopicName	String	0..1	attr	Name of the DDS Request Topic associated with the Field.

△

Class	**DdsServiceInterfaceDeployment**			
methodReply TopicName	String	0..1	attr	Name of the DDS Reply Topic associated with the Method.
methodRequest TopicName	String	0..1	attr	Name of the DDS Request Topic associated with the Method.
serviceInterface Id	String	1	attr	Unique Identifier that identifies the ServiceInterface in DDS. This Identifier is encoded in the USER_DATA QoS of the DomainParticipant associated with the Service Instance and its value is propagated by DDS Discovery messages. **Tags:**atp.Status=draft
transport Protocol	String	*	attr	This attribute defines over which Transport Layer Protocol(s) this Method is intended to be sent.

Table 10.12: DdsServiceInterfaceDeployment

[TPS_MANI_03556]{DRAFT} **DDS-RPC Service Binding** ⌈The `DdsServiceInterfaceDeployment` meta-class provides the ability to configure the name of the DDS Request and Reply Topics associated with a DDS-RPC Service with the `methodRequestTopicName` and `methodReplyTopicName` attributes, respectively. DDS-RPC Services are the mechanisms specified in the OMG RPC over DDS specification (DDS-RPC [26]) to handle method calls with DDS.

The `methodRequestTopicName` and `methodReplyTopicName` attributes are optional, if unspecified they shall be configured as specified in the DDS-RPC specification.⌋*(RS_MANI_00038)*

[TPS_MANI_03526]{DRAFT} **DDS `VariableDataPrototype` binding** ⌈The `DdsEventDeployment` meta-class provides the ability to bind a `VariableDataPrototype` to DDS and to assign a DDS Topic to the `event` with the `topicName` attribute. Moreover, the meta-class provides the ability to configure the `transportProtocol`s over which the `VariableDataPrototype` may be accessed.⌋*(RS_MANI_00038)*

Class	**DdsEventDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	DDS configuration settings for an Event. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceEventDeployment*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
topicName	String	0..1	attr	Name of the DDS Topic associated with the Event. **Tags:**atp.Status=draft
transport Protocol	String	*	attr	This attribute defines over which Transport Layer Protocol(s) this event is intended to be sent. **Tags:**atp.Status=draft

Table 10.13: DdsEventDeployment

[TPS_MANI_03557]{DRAFT} **DDS `ClientServerOperation` Binding** ⌈There exists no concrete subclass of `ServiceMethodDeployment` to bind a

ClientServerOperation to DDS. This binding is done with the DdsServiceInterfaceDeployment (see [TPS_MANI_03556]).⌋(*RS_MANI_00038*)

[TPS_MANI_03558]{DRAFT} **DDS Field Binding** ⌈The DdsFieldDeployment meta-class provides the ability to bind a Field to DDS.

To bind the Field's notification event the notifier is used to define a DDS Topic. To assign the get/set Field methods the fieldRequestTopicName and fieldReplyTopicName are used to define a DDS Topic.

The fieldRequestTopicName and fieldReplyTopicName attributes are optional, if unspecified they shall be configured as specified in the DDS-RPC specification.⌋(*RS_MANI_00038*)

[TPS_MANI_03622]{DRAFT} **DDS Transport Protocols are up to the stack implementer** ⌈Underlying transports below the RTPS protocol are not part of the DDS OMG standard (QoS APIs [25]) or (XML schema [27]). It is up to each DDS implementation vendor to decide which transports are supported and how those are expressed in APIs and XML.⌋(*RS_MANI_00038*)

Class	**DdsFieldDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	DDS configuration settings for a Field. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceFieldDeployment*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
notifier	DdsEventDeployment	0..1	aggr	This aggregation represents the settings of the notifier. **Tags:**atp.Status=draft

Table 10.14: DdsFieldDeployment

[constr_3563]{DRAFT} **Mandatory topic name values** ⌈The attributes methodRequestTopicName, methodReplyTopicName, fieldRequestTopicName, fieldReplyTopicName, topicName shall specify string values, each of them unique within the service interface.⌋()

10.1.3 User Defined Service Interface

This chapter describes a user defined deployment of a ServiceInterface to a middleware technology that is not standardized by AUTOSAR. Such UserDefinedServiceInterfaceDeployment can for example also be used to describe a machine local IPC communication.

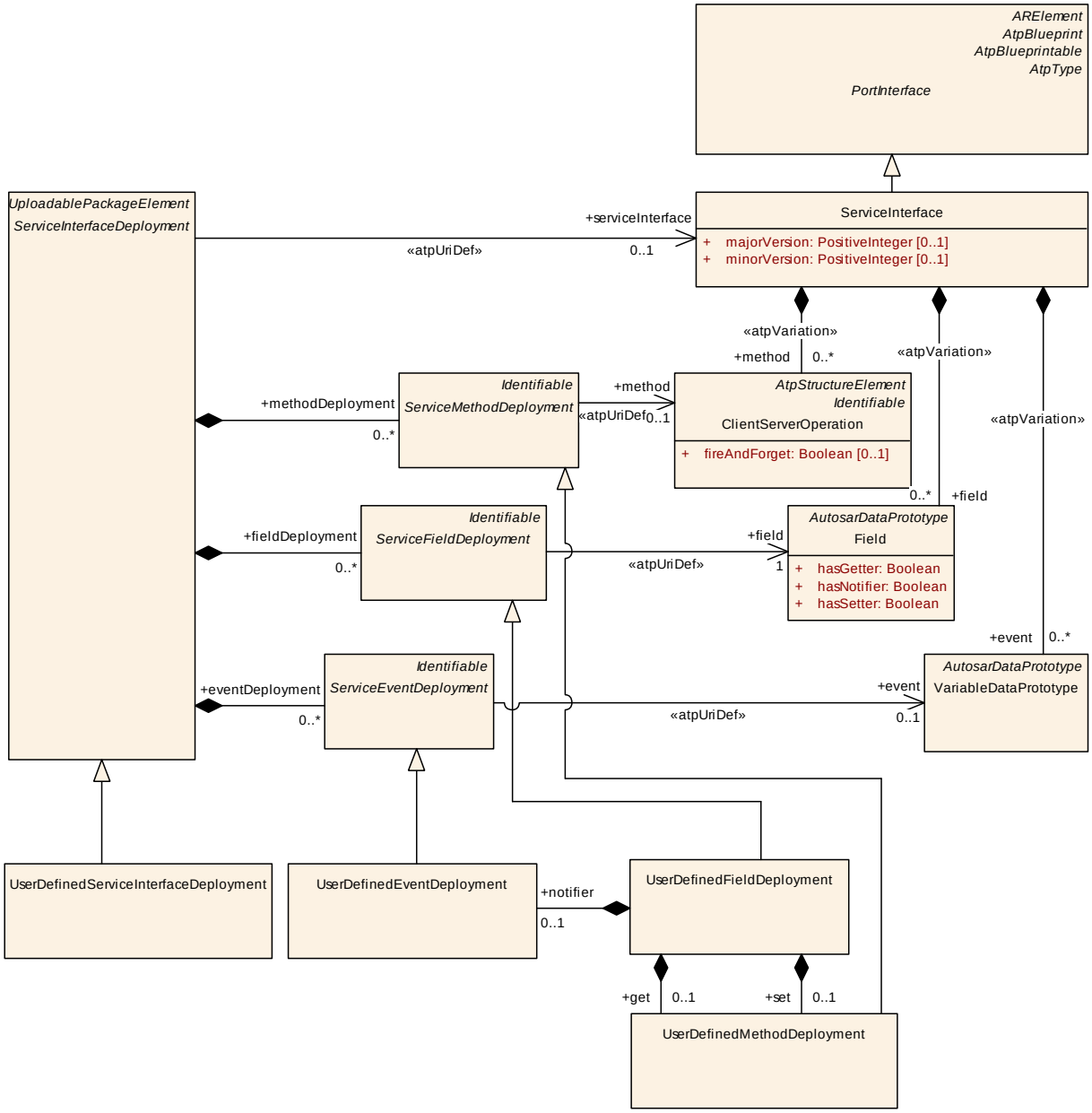


Figure 10.5: User defined deployment of ServiceInterface

[TPS_MANI_03045]{DRAFT} **UserDefined `ServiceInterface` binding** ⌈The `UserDefinedServiceInterfaceDeployment` meta-class provides the ability to bind a `ServiceInterface` that is referenced in the role `serviceInterface` to a middleware technology that is not standardized by AUTOSAR.⌋*(RS_MANI_00014)*

Please note that `UserDefinedServiceInterfaceDeployment` is `Identifiable` and therefore it is able to describe special data (sdg) which is not represented by the standard model.

Class	**UserDefinedServiceInterfaceDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	UserDefined configuration settings for a ServiceInterface. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInterfaceDeployments			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *ServiceInterfaceDeployment*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 10.15: UserDefinedServiceInterfaceDeployment

[TPS_MANI_01165]{DRAFT} **Standardized value of `UserDefinedServiceInterfaceDeployment.category`** ⌈The AUTOSAR Standard reserves the following value for attribute `UserDefinedServiceInterfaceDeployment.category`:

- `SERVICE_INTERFACE_DEPLOYMENT_IPC`

It is possible to use a custom, non-standardized value for the attribute `UserDefinedServiceInterfaceDeployment.category` but this option comes with the obligation to use a value that is guaranteed to not clash with possible future extensions of the collection of standardized values.⌋*(RS_MANI_00014)*

IPC communication may or may not require configuration settings that nevertheless aren't standardized by AUTOSAR. The best support that the AUTOSAR standard can deliver is the provision of meta-classes that can be taken as the basis to define configuration settings by means of the definition of `Sdg`.

[constr_1570]{DRAFT} **Restriction for `UserDefinedServiceInterfaceDeployment` of `category` `SERVICE_INTERFACE_DEPLOYMENT_IPC`** ⌈An `AdaptivePlatformServiceInstance` that references a `UserDefinedServiceInterfaceDeployment` of `category` `SERVICE_INTERFACE_DEPLOYMENT_IPC` shall **only** be referenced by a `UserDefinedServiceInstanceToMachineMapping` in the role `serviceInstance` that in turn references a `UserDefinedCommunicationConnector`.⌋*()*

Rationale for [constr_1570]: for a local IPC binding it is sometimes necessary to define properties of the IPC system. And for this purpose the `UserDefinedCommunicationConnector` mapped to an `AdaptivePlatformServiceInstance` can be used to define global properties (e.g. for service discovery) of a given "IPC-Domain".

In other words, each defined `UserDefinedCommunicationConnector` may represent such an "IPC-Domain" that requires a dedicated configuration on the basis of the definition of `Sdg`s.

Class	UserDefinedCommunicationConnector			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::CddSupport			
Note	This element allows the modeling of arbitrary Communication Connectors.			
Base	*ARObject*, *CommunicationConnector*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 10.16: UserDefinedCommunicationConnector

[TPS_MANI_03046]{DRAFT} **User defined `VariableDataPrototype` binding** ⌈The `UserDefinedEventDeployment` meta-class provides the ability to bind a `VariableDataPrototype` that is referenced in the role `event` to a middleware technology that is not standardized by AUTOSAR.⌋*(RS_MANI_00014)*

Please note that `UserDefinedEventDeployment` is `Identifiable` and therefore it is able to describe special data (sdg) which is not represented by the standard model.

Class	UserDefinedEventDeployment			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	UserDefined configuration settings for an Event. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceEventDeployment*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 10.17: UserDefinedEventDeployment

[TPS_MANI_03047]{DRAFT} **User defined `ClientServerOperation` binding** ⌈The `UserDefinedMethodDeployment` meta-class provides the ability to bind a `ClientServerOperation` that is referenced in the role `method` to a middleware technology that is not standardized by AUTOSAR.⌋*(RS_MANI_00014)*

Please note that `UserDefinedMethodDeployment` is `Identifiable` and therefore it is able to describe special data (sdg) which is not represented by the standard model.

Class	UserDefinedMethodDeployment			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	UserDefined configuration settings for a Method. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceMethodDeployment*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 10.18: UserDefinedMethodDeployment

[TPS_MANI_03048]{DRAFT} **User defined `Field` binding** ⌈The `UserDefinedFieldDeployment` meta-class provides the ability to bind a `Field` that is referenced

in the role field to a middleware technology that is not standardized by AUTOSAR.⌋ *(RS_MANI_00014)*

Please note that UserDefinedFieldDeployment is Identifiable and therefore it is able to describe special data (sdg) which is not represented by the standard model.

Class	**UserDefinedFieldDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	UserDefined configuration settings for a Field. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceFieldDeployment*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
get	UserDefinedMethod Deployment	0..1	aggr	This aggregation represents the settings of the get method **Tags:**atp.Status=draft
notifier	UserDefinedEvent Deployment	0..1	aggr	This aggregation represents the settings of the notifier. **Tags:**atp.Status=draft
set	UserDefinedMethod Deployment	0..1	aggr	This aggregation represents the settings of the set method **Tags:**atp.Status=draft

Table 10.19: UserDefinedFieldDeployment

[constr_3417]{DRAFT} **UserDefinedEventDeployments aggregated by a UserDefinedFieldDeployment** ⌈A UserDefinedEventDeployment that is aggregated by a UserDefinedFieldDeployment in the role notifier shall not reference a VariableDataPrototype in the role event.⌋ *()*

[constr_3418]{DRAFT} **UserDefinedMethodDeployments aggregated by a UserDefinedFieldDeployment** ⌈A UserDefinedMethodDeployment that is aggregated by a UserDefinedFieldDeployment in the role get or set shall not reference a ClientServerOperation in the role method.⌋ *()*

10.2 Service Instance Deployment

An AdaptivePlatformServiceInstance makes the functionality of a ServiceInterface available on the *AUTOSAR adaptive platform*. Several AdaptivePlatformServiceInstances may be set up for the same ServiceInterface. They deliver the same functionality, but for different purposes and/or to different users.

The ProvidedApServiceInstance represents a provider that offers the functionality of a ServiceInterface with particular properties. Clients that are represented by the RequiredApServiceInstance observe offers and choose a provider with respect to service properties.

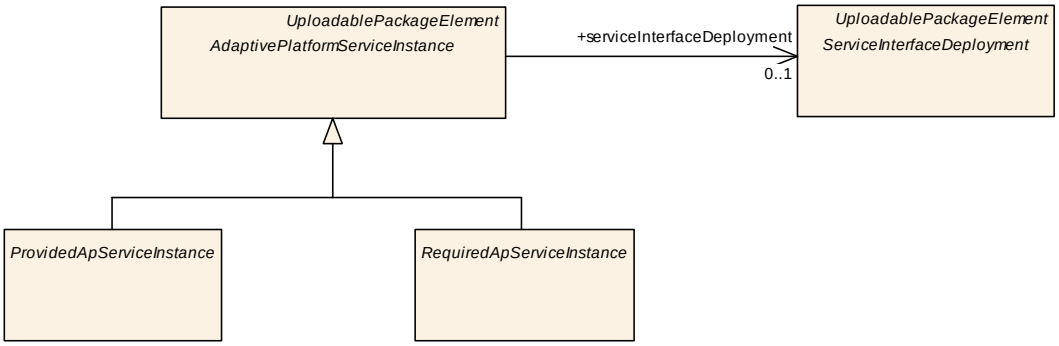


Figure 10.6: Modeling of the `AdaptivePlatformServiceInstance`

Note that the abstract meta-class `AdaptivePlatformServiceInstance` is derived from `ARElement`. This means that all meta-classes derived from `AdaptivePlatformServiceInstance` can be declared on the M1 level as part of an `ARPackage` and thus can be used in several Manifest descriptions.

Class	***AdaptivePlatformServiceInstance*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a service instance in an abstract way. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Subclasses	*ProvidedApServiceInstance*, *RequiredApServiceInstance*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
e2eEvent ProtectionProps	End2EndEvent ProtectionProps	*	aggr	This aggregation allows to protect an event or a field notifier that is defined inside of the ServiceInterface that is referenced by the ServiceInstance in the role service Interface. **Tags:**atp.Status=draft
e2eMethod ProtectionProps	End2EndMethod ProtectionProps	*	aggr	This aggregation allows to protect a method or a field getter or a field setter that is defined inside of the Service Interface that is referenced by the ServiceInstance in the role serviceInterface **Tags:**atp.Status=draft
secureCom Config	ServiceInterface ElementSecureCom Config	*	aggr	Configuration settings to secure the communication of ServiceInterface elements. **Tags:**atp.Status=draft
serviceInterface Deployment	ServiceInterface Deployment	0..1	ref	Reference to a ServiceInterfaceDeployment that identifies the ServiceInterface that is represented by the Service Instance. **Tags:**atp.Status=draft

Table 10.20: AdaptivePlatformServiceInstance

Class	***RequiredApServiceInstance*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in an abstract way. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Subclasses	DdsRequiredServiceInstance, RequiredSomeipServiceInstance, RequiredUserDefinedServiceInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 10.21: RequiredApServiceInstance

Class	***ProvidedApServiceInstance*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in an abstract way. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Subclasses	DdsProvidedServiceInstance, ProvidedSomeipServiceInstance, ProvidedUserDefinedServiceInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 10.22: ProvidedApServiceInstance

There are two alternative ways to relate an `AdaptivePlatformServiceInstance` with a `MachineDesign` as described in [TPS_MANI_03000] and [TPS_MANI_03001]. Figure Figure 10.7 shows both approaches in an example.

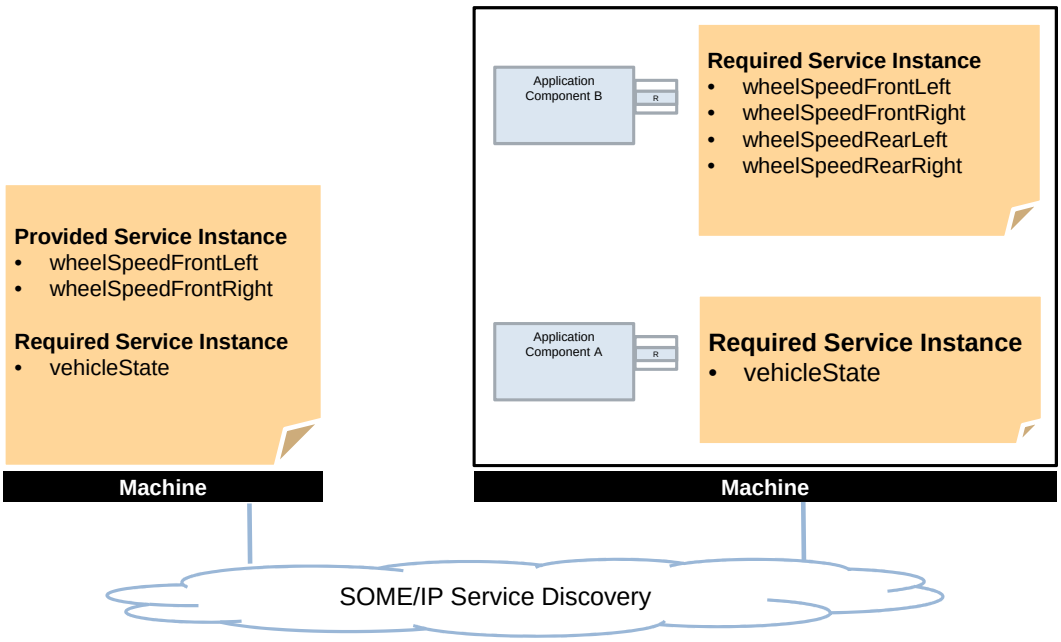


Figure 10.7: Different approaches for ServiceInstanceMapping

[TPS_MANI_03001]{DRAFT} **Mapping of `AdaptivePlatformServiceInstance` to a `MachineDesign`** ⌈`ServiceInstanceToMachineMapping` is used to assign one or several `AdaptivePlatformServiceInstance`s to (via a `Communication-Connector`) a `MachineDesign`. This allows to define a “black box” machine view without any assumption on the application software but with all necessary information to configure the communication (e.g. SOME/IP).⌋*(RS_MANI_00009)*

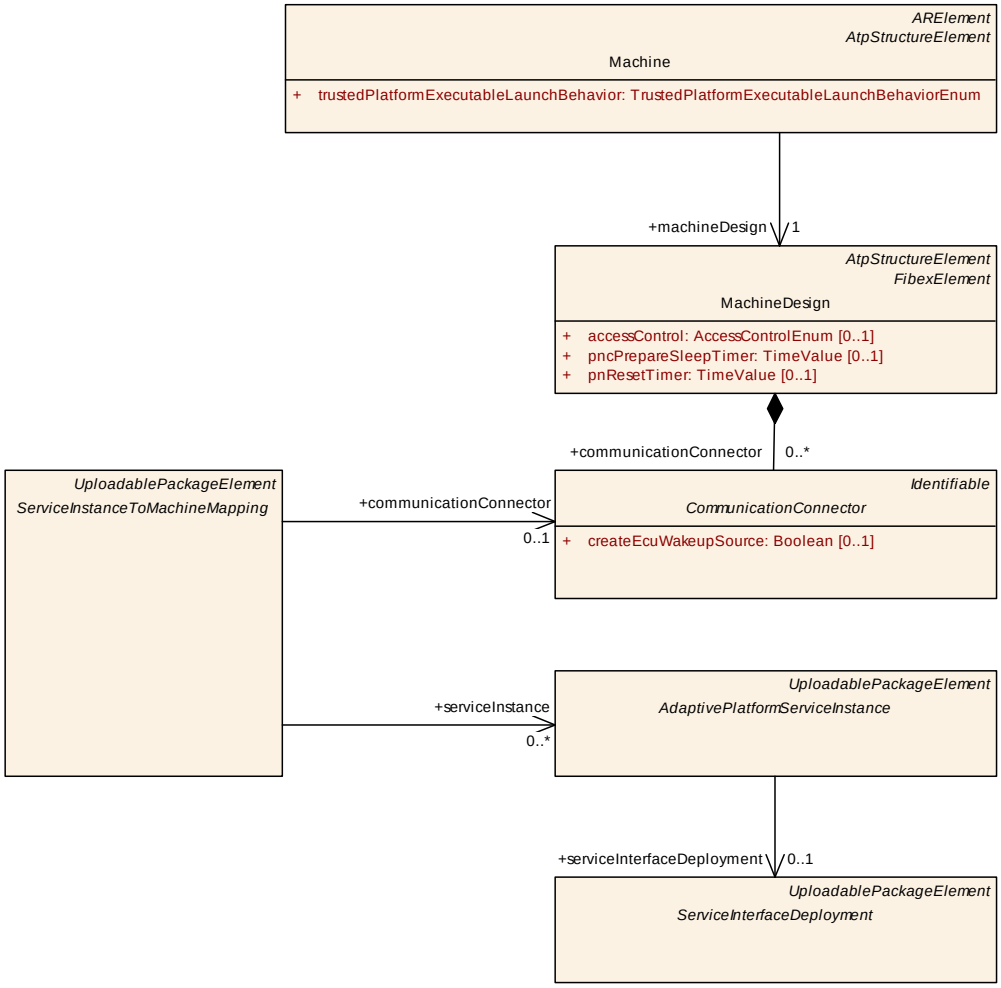


Figure 10.8: ServiceInstanceToMachineMapping

Class	***ServiceInstanceToMachineMapping*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping
Note	This meta-class represents the ability to map one or several AdaptivePlatformServiceInstances to a CommunicationConnector of a Machine. **Tags:**atp.Status=draft
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*
Subclasses	DdsServiceInstanceToMachineMapping, SomeipServiceInstanceToMachineMapping, UserDefined ServiceInstanceToMachineMapping

▽

△

Class	*ServiceInstanceToMachineMapping* (abstract)			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Connector	Communication Connector	0..1	ref	Reference to the Machine to which the ServiceInstance is mapped. **Tags:**atp.Status=draft
secOcCom PropsFor Multicast	SecOcSecureCom Props	*	ref	Reference to communication security configuration settings that are valid for the udp multicast endpoint (Port + Multicast IP Address) defined by the ServiceInstanceTo MachineMapping. **Tags:**atp.Status=draft
secureCom PropsForTcp	SecureComProps	*	ref	Reference to communication security configuration settings that are valid for the tcp unicast endpoint (Tcp Port + Unicast IP Address) defined by the Service InstanceToMachineMapping. **Tags:**atp.Status=draft
secureCom PropsForUdp	SecureComProps	*	ref	Reference to communication security configuration settings that are valid for the udp unicast endpoint (Udp Port + Unicast IP Address) defined by the Service InstanceToMachineMapping. **Tags:**atp.Status=draft
serviceInstance	AdaptivePlatform ServiceInstance	*	ref	Reference to a ServiceInstance that is mapped to the Machine. **Tags:**atp.Status=draft

Table 10.23: ServiceInstanceToMachineMapping

[constr_5155]{DRAFT} `SomeipServiceInstanceToMachineMapping` **only supports a single Address Family** ⌈A `SomeipServiceInstanceToMachineMapping` shall only support a single Address Family, i.e. either IPv4 or IPv6. The address family shall be consistent with the `Ipv4Configuration`/`Ipv6Configuration` of the `NetworkEndpoint` referenced by the `EthernetCommunicationConnector` that is referenced by the `SomeipServiceInstanceToMachineMapping` in the role `communicationConnector`.⌋*()*

[constr_3487]{DRAFT} **TCP endpoint can only serve provided or required service instances exclusively** ⌈`ServiceInstanceToMachineMapping` is not allowed to refer to a `ProvidedApServiceInstance` and at the same time a `RequiredApServiceInstance` in the role `serviceInstance` if

- the `ServiceInterfaceDeployment` that is referenced by the `ProvidedApServiceInstance` in the role `serviceInterfaceDeployment` and
- the `ServiceInterfaceDeployment` that is referenced by the `RequiredApServiceInstance` in the role `serviceInterfaceDeployment`

both contain defined tcp content that is described by the transportProtocol attribute in the deployment elements of SOME/IP or DDS.

In other words a TCP endpoint can only serve provided or required service instances exclusively.⌋*()*

The reason for [constr_3487] is that the POSIX Socket API does not support the binding of several TCP sockets onto the same tuple <local IP address, local port>. But this would be necessary if a service is provided and consumed over the same TCP Endpoint.

[TPS_MANI_03000]{DRAFT} **Mapping of `AdaptivePlatformServiceInstance` to `PortPrototype`s** ⌈`ServiceInstanceToPortPrototypeMapping` is used to assign an `AdaptivePlatformServiceInstance` to a `PortPrototype` of a `SwComponentType`. This allows to define how specific `PortPrototype`s of a Software Component are represented in the middleware in terms of the service configuration.⌋*(RS_MANI_00011)*

In other words, the “outside” appearance of a `PortPrototype` from the middleware point of view is the `AdaptivePlatformServiceInstance`, or the concrete subclasses `RequiredApServiceInstance` and `ProvidedApServiceInstance`.

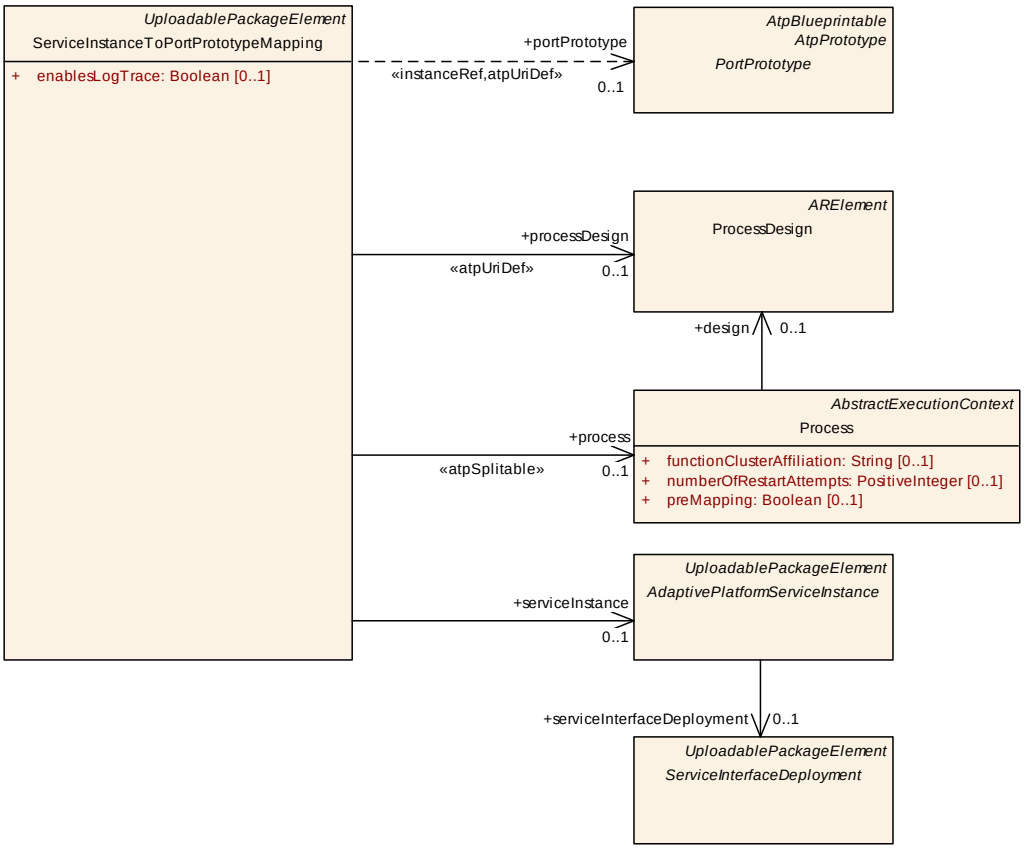


Figure 10.9: ServiceInstanceToPortPrototypeMapping

Class	**ServiceInstanceToPortPrototypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	This meta-class represents the ability to assign a transport layer dependent ServiceInstance to a Port Prototype. With this mapping it is possible to define how specific PortPrototypes are represented in the middleware in terms of service configuration. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInstanceToPortPrototypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
enablesLog Trace	Boolean	0..1	attr	This attribute enables/disables Log&Trace for the communication on the referenced Port of the referenced process. True: Log&Trace is enabled. False: Log&Trace is disabled.
portPrototype	PortPrototype	0..1	iref	Reference to a specific PortPrototype that represents the ServiceInstance. **Tags:**atp.Status=draft **InstanceRef implemented by:**PortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	Reference to the Process in which the enclosing Service InstanceToPortPrototypeMapping is executed. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
processDesign	ProcessDesign	0..1	ref	Reference to the ProcessDesign in which the Executable that contains the SoftwareComponent and the referenced PortPrototype is executed. **Stereotypes:** atpUriDef **Tags:**atp.Status=draft
serviceInstance	AdaptivePlatform ServiceInstance	0..1	ref	Reference to a ServiceInstance that is represented in the Software Component by the mapped group of Port Prototypes. **Tags:**atp.Status=draft

Table 10.24: ServiceInstanceToPortPrototypeMapping

Meta-classes `ProvidedApServiceInstance` and `RequiredApServiceInstance` are abstract and this allows for using specific derived classes that fit the underlying middleware (e.g. SOME/IP). The following sub-chapters will detail the supported specializations.

[TPS_MANI_01316]{DRAFT} **Existence of `ServiceInstanceToPortPrototypeMapping.processDesign`** ⌈The reference `ServiceInstanceToPortPrototypeMapping.processDesign` shall only be used in a design-level modeling scenario where a pre-assignment of a given `ServiceInstanceToPortPrototypeMapping` to a specific `ProcessDesign` is intended.

By this means it is possible to express that one `Executable` is foreseen to be executed in multiple instances and it is also possible to assign service instances to each of the foreseen `ProcessDesign`s that represent instances of `Executable` at design time.⌋*(RS_MANI_00009)*

[TPS_MANI_01317]{DRAFT} **Existence of ServiceInstanceToPortPrototypeMapping.process** ⌈The reference ServiceInstanceToPortPrototypeMapping.process shall be only used in a deployment-level modeling where the integration of a SoftwareCluster is created. The reference has the role to identify the actual Process used in the execution manifest.

This reference overwrites a potentially existing reference to a ProcessDesign in the context of the enclosing ServiceInstanceToPortPrototypeMapping⌋*(RS_MANI_00009)*

Please note that if both ServiceInstanceToPortPrototypeMapping.processDesign and process exist, the latter gets the higher significance because it is created by an integrator who may overrule design decisions on the basis of superior knowledge about the context.

In such a case it is acceptable that the Process references a different ProcessDesign than the one referenced in the role ServiceInstanceToPortPrototypeMapping.processDesign. This is just the result of superior knowledge of the integrator over the designer.

10.2.1 SOME/IP Service Instance Deployment

In the case of SOME/IP used as the middleware the derived meta-classes are ProvidedSomeipServiceInstance or RequiredSomeipServiceInstance. These meta-classes also carry attributes that apply for the service discovery on SOME/IP.

Primitive	**AnyServiceInstanceId**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This is a positive integer or the literal ALL (the value ANY is technically supported but deprecated) which can be denoted in decimal, octal and hexadecimal. The value is between 0 and 4294967295. **Tags:** xml.xsd.customType=ANY-SERVICE-INSTANCE-ID xml.xsd.pattern=[1-9][0-9]*|0[xX][0-9a-fA-F]+|0[0-7]*|0[bB][0-1]+|ANY|ALL xml.xsd.type=string

Table 10.25: AnyServiceInstanceId

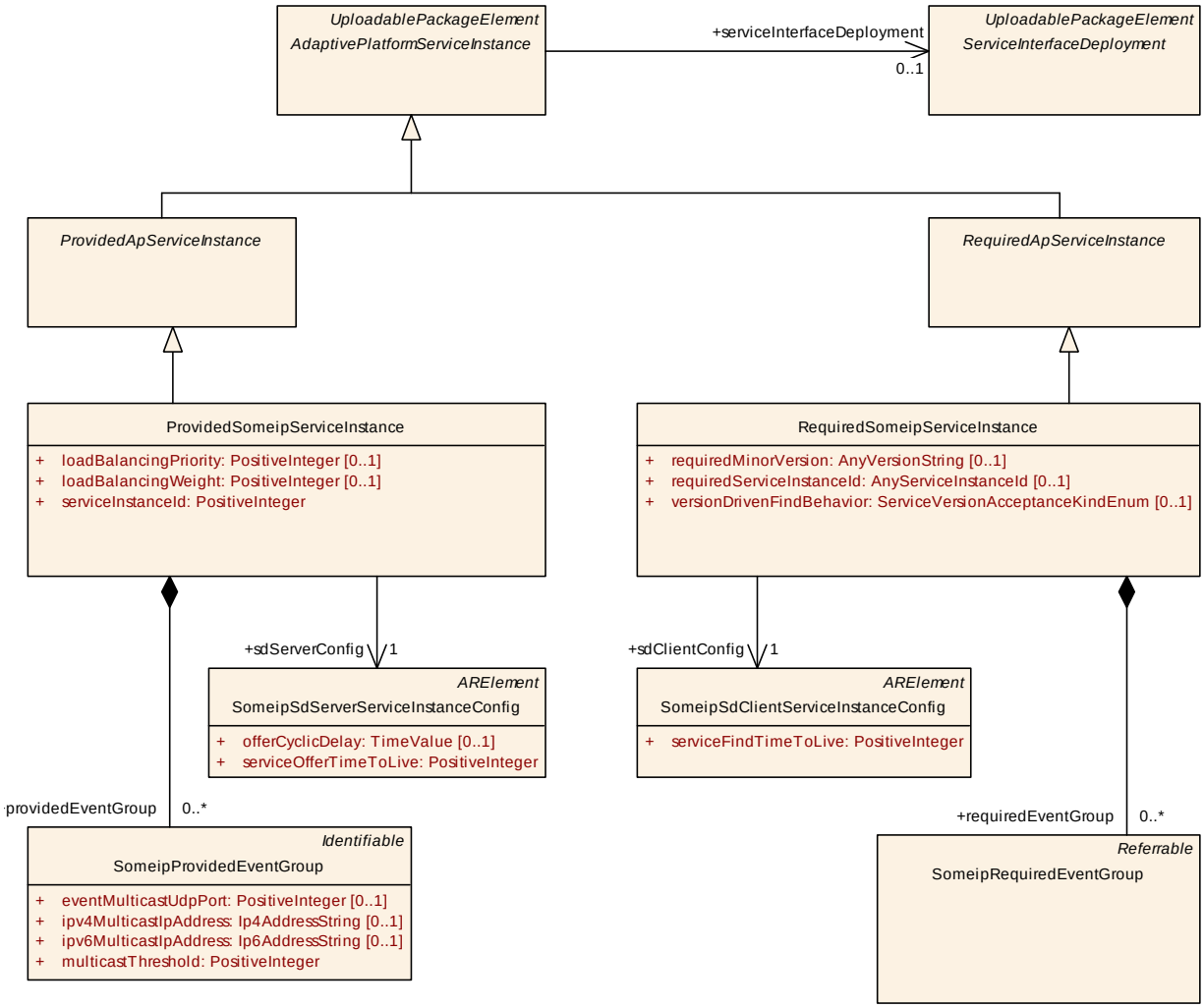


Figure 10.10: SOME/IP Service Instances

10.2.1.1 Provided Service Instance

The `ProvidedSomeipServiceInstance` defines the `serviceInstanceId` for the Service Instance of the `SomeipServiceInterfaceDeployment` that is referenced with the `serviceInterfaceDeployment` reference.

It means that the Server on which the `ProvidedSomeipServiceInstance` is deployed offers the Service Instance over SOME/IP with the `serviceInstanceId` and `serviceInterfaceId`.

Class	**ProvidedSomeipServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in a concrete implementation on top of SOME/IP. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInstances			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *ProvidedApServiceInstance*, *Referrable*, *Uploadable PackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
capability Record (ordered)	TagWithOptionalValue	*	aggr	A sequence of records to store arbitrary name/value pairs conveying additional information about the named service. **Tags:**atp.Status=draft
eventProps	SomeipEventProps	*	aggr	Configuration settings for individual events that are provided by the ServiceInstance. **Tags:**atp.Status=draft
loadBalancing Priority	PositiveInteger	0..1	attr	This attribute is used to specify the priority in the load balancing option of SOME/IP that is added to the Offer Service. When a client searches for all service instances of a service, the client shall choose the service instance with highest priority if one is defined.
loadBalancing Weight	PositiveInteger	0..1	attr	This attribute is used to specify the weight in the load balancing option of SOME/IP that is added to the Offer Service. When a client searches for all service instances of a service, the client shall choose the service instance with highest priority if one is defined. If several service instances exist with the highest priority the service instance shall be chosen based on the weights of the service instances.
method ResponseProps	SomeipMethodProps	*	aggr	Configuration settings for individual methods that are provided by the ServiceInstance. **Tags:**atp.Status=draft
providedEvent Group	SomeipProvidedEvent Group	*	aggr	List of EventGroups that are provided by the Service Instance. **Tags:**atp.Status=draft
sdServerConfig	SomeipSdServer ServiceInstanceConfig	1	ref	Server specific configuration settings relevant for the SOME/IP service discovery. **Tags:**atp.Status=draft
serviceInstance Id	PositiveInteger	1	attr	Identification number that is used by SOME/IP service discovery to identify the instance of the service. The value 4294967295 for service instance id is reserved and should not be used.

Table 10.26: ProvidedSomeipServiceInstance

[constr_3287]{DRAFT} **Mandatory information of a `ProvidedSomeipServiceInstance`** ⌈The `ProvidedSomeipServiceInstance` shall always define the `serviceInstanceId`.⌋*()*

[constr_1770]{DRAFT} **Value of `ProvidedSomeipServiceInstance.serviceInstanceId`** ⌈For each `ProvidedSomeipServiceInstance.serviceInstanceId`, the value 4294967295 shall not be used.⌋*()*

Rationale for [constr_1770]: on protocol level, the value 4294967295 represents the "ALL" semantics that can only be used in find messages.

In addition to the service identification properties a SOME/IP offer message contains so called endpoint options that define how the service instance is reachable by clients.

[TPS_MANI_03168]{DRAFT} **Configuration of the SOME/IP load balancing option** ⌈The SOME/IP load balancing option is configurable per `ProvidedSomeipServiceInstance` with the two attributes `loadBalancingPriority` and `loadBalancingWeight`.⌋*(RS_MANI_00024)*

The SOME/IP load balancing option is used to prioritize different `ProvidedSomeipServiceInstance`s that point to the same `SomeipServiceInterfaceDeployment`, so that a client chooses the service instance based on these settings. This option is attached to SOME/IP Offer Service entries.

[constr_3415]{DRAFT} **Value range of `loadBalancingPriority`** ⌈The value of `loadBalancingPriority` shall be in the range of 0..65535.⌋*()*

Please note that according to SOME/IP a lower value means higher priority.

[constr_3416]{DRAFT} **Value range of `loadBalancingWeight`** ⌈The value of `loadBalancingWeight` shall be in the range of 0..65535.⌋*()*

Please note that according to SOME/IP a higher value means higher probability to be chosen.

[constr_1723]{DRAFT} **`ProvidedSomeipServiceInstance` shall be unique in respect of `serviceInstanceId`, `serviceInterfaceId` and `majorVersion` on a VLAN** ⌈On a VLAN, each `ProvidedSomeipServiceInstance` shall have a different `serviceInstanceId`, `serviceInterfaceId` and `majorVersion` value combination.

In other words, no two `ProvidedSomeipServiceInstance`s shall have the same `serviceInstanceId`, `serviceInterfaceId` and `majorVersion` value combination during runtime on the same VLAN.⌋*()*

The following figure shows that different SOME/IP `ServiceInstance`s with the same `serviceInstanceId`, `serviceInterfaceId` and `majorVersion` are provided on different VLANs. This is a valid setup according to [constr_1723].

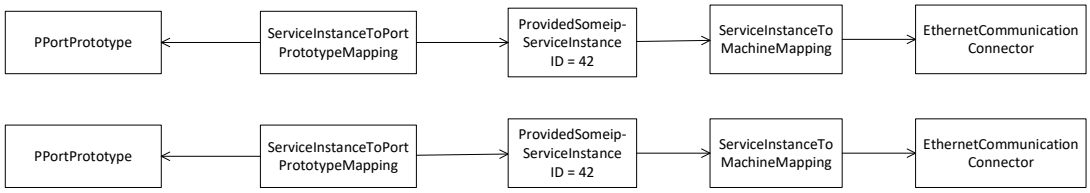


Figure 10.11: Scenario in which two `ProvidedSomeipServiceInstance`s with the same credentials are provided on two VLANs

In the following example where the same `ProvidedSomeipServiceInstance` is mapped to different `PPortPrototype`s and is provided on different VLANs the specification item [TPS_MANI_03236] applies. This means that only one of the `PPortPrototype`s is active at runtime at the same time and offers the `ServiceInstance` on two different VLANs.

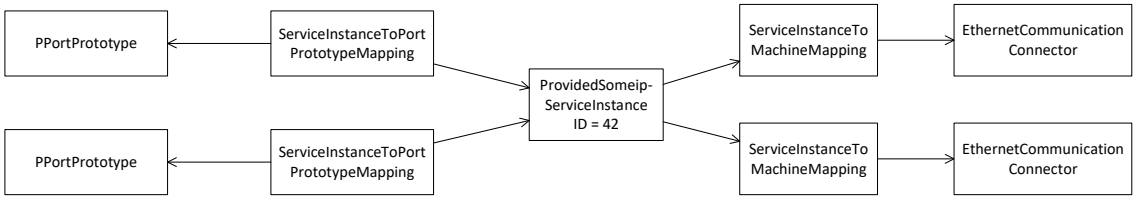


Figure 10.12: Scenario in which a `ProvidedSomeipServiceInstance` is provided on two VLANs and is mapped to different `PPortPrototype`s from which latest at runtime only one is operational

[TPS_MANI_03236]{DRAFT} **Mapping of `ProvidedSomeipServiceInstance` to different `PPortPrototype`s** ⌈In case that the same `ProvidedSomeipServiceInstance` is mapped by several `ServiceInstanceToPortPrototypeMapping`s to different `PPortPrototype`s it shall be ensured (latest at runtime) that only one of these mapped `PPortPrototype`s is actually operational at any given point in time.⌋*()*

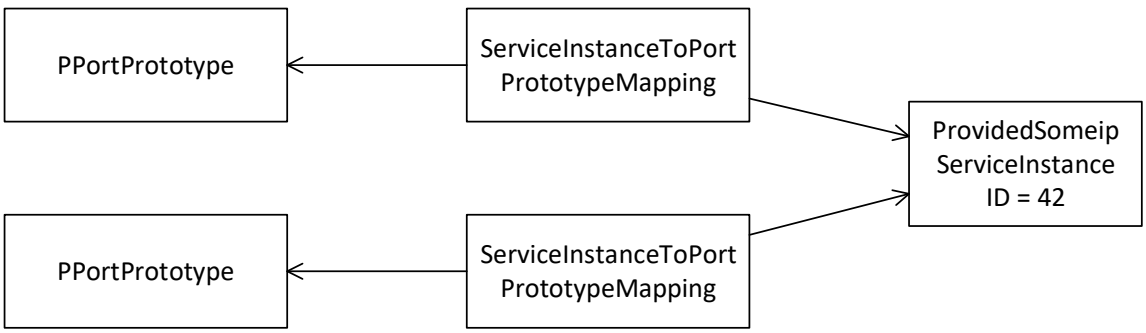


Figure 10.13: Static setup in which a `ProvidedSomeipServiceInstance` is mapped to two `PPortPrototype`s from which latest at runtime only one is operational

Please note that two ProvidedSomeipServiceInstance elements with the same credentials according to [constr_1723] may exist that both are mapped to different PPortPrototypes and the ProvidedSomeipServiceInstances are mapped by ServiceInstanceToMachineMappings to the same VLAN. At runtime only one of these PPortPrototypes shall be operational at any given point in time.

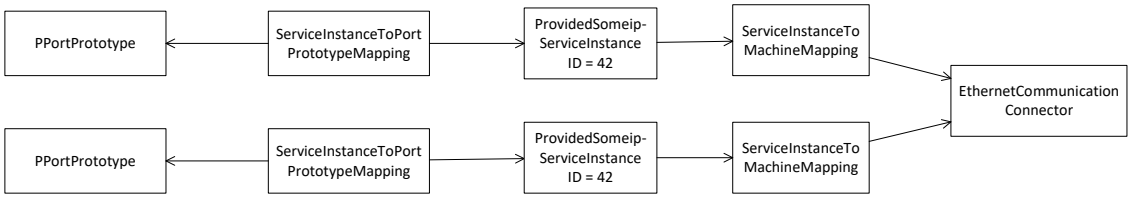


Figure 10.14: Static setup in which ProvidedSomeipServiceInstances with the same credentials provided on the same VLAN are mapped to two different PPortPrototypes from which latest at runtime only one is operational

Such a scenario may be created by the installation of two separate SoftwareClusters as shown in the following figure. It is not possible to check such a setup since the SoftwareClusters may be developed, integrated, and deployed independent from each other.

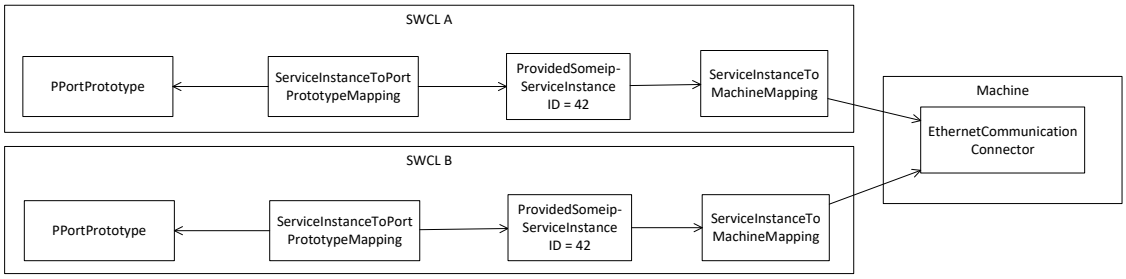


Figure 10.15: SoftwareCluster scenario in which ProvidedSomeipServiceInstances with the same credentials provided on the same VLAN exist

10.2.1.1.1 IP Configuration

In SOME/IP the Offer service entry references IPv4 or IPv6 Endpoint options to indicate to the client where the server accepts the method calls and where the server sends the event messages.

Such an Endpoint contains the IP address of the sender. The IP address configuration is described in this chapter.

[TPS_MANI_03002]{DRAFT} **IP configuration for a ProvidedSomeipServiceInstance** ⌈A ProvidedSomeipServiceInstance can be mapped to a CommunicationConnector of a MachineDesign with the SomeipServiceInstanceToMachineMapping.

With this mapping an assignment of the ProvidedSomeipServiceInstance to a unicast IP Address is established since the EthernetCommunicationConnector refers to a NetworkEndpoint in the role unicastNetworkEndpoint.⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03003]{DRAFT} **ProvidedSomeipServiceInstance Fanout** ⌈It is allowed to map the same ProvidedSomeipServiceInstance to different CommunicationConnectors of a MachineDesign. In such a case, several SomeipServiceInstanceToMachineMappings shall be defined.

This allows for offering the same ProvidedSomeipServiceInstance on different VLANs or even on different CommunicationClusters.⌋*(RS_MANI_00009, RS_MANI_00024)*

[constr_3538]{DRAFT} **Only one ServiceInstanceToMachineMapping per technology and CommunicationConnector** ⌈Each AdaptivePlatformServiceInstance shall only be referenced up to once by a specific ServiceInstanceToMachineMapping subclass in the role serviceInstance where the ServiceInstanceToMachineMapping refer to the same CommunicationConnector.⌋*()*

In other words, it is not allowed to define for the same service instance two ServiceInstanceToMachineMapping of the same kind (e.g. SomeipServiceInstanceToMachineMapping) which refer to the same CommunicationConnector.

[TPS_MANI_03554]{DRAFT} **Several SomeipServiceInstanceToMachineMappings with equal settings** ⌈If

- one SomeipServiceInstanceToMachineMapping refers to several service instances in the role serviceInstance
- several SomeipServiceInstanceToMachineMappings with equal settings refer to several service instances in the role serviceInstance
- the combination of the two above applies

then for all the referenced service instances the same network connection (i.e. Ethernet socket) will be used.⌋*(RS_MANI_00009, RS_MANI_00024)*

[constr_5052]{DRAFT} **ProvidedSomeipServiceInstances of the same serviceInterface on one Machine** ⌈Different ProvidedSomeipServiceInstances referring to the same SomeipServiceInterfaceDeployment shall not be mapped by a SomeipServiceInstanceToMachineMapping to the same IPAddress defined in the NetworkEndpoint that is referenced by the EthernetCommunicationConnector.unicastNetworkEndpoint and to the same port number represented by either SomeipServiceInstanceToMachineMapping.udpPort or SomeipServiceInstanceToMachineMapping.tcpPort.⌋*()*

The reason for this restriction is that the Instance IDs are only used for Service Discovery but are not contained in the SOME/IP header. So if for example two ProvidedSomeipServiceInstances of the same ServiceInterface are provided on the same machine and a client wants to call a method of one of these ProvidedSomeipServiceInstances the only possibility for the client to distinguish the ProvidedSomeipServiceInstances is the port number over which the individual ProvidedSomeipServiceInstances are provided.

[TPS_MANI_03555]{DRAFT} **Mix of SomeipServiceInstanceToMachineMapping and signal-based communication** ⌈SomeipServiceInstanceToMachineMapping defines service instance communication on a specific Ethernet socket and the same socket may also be used for signal-based communication at the same time.⌋ *(RS_MANI_00009, RS_MANI_00024)*

Please note that the signal-based communication is described in section 11.

Via the definition of respective ISignalTriggering, PduTriggering, and SocketConnection for signal-based communication, the same values for Ethernet address and port may be defined as used at the SomeipServiceInstanceToMachineMapping.

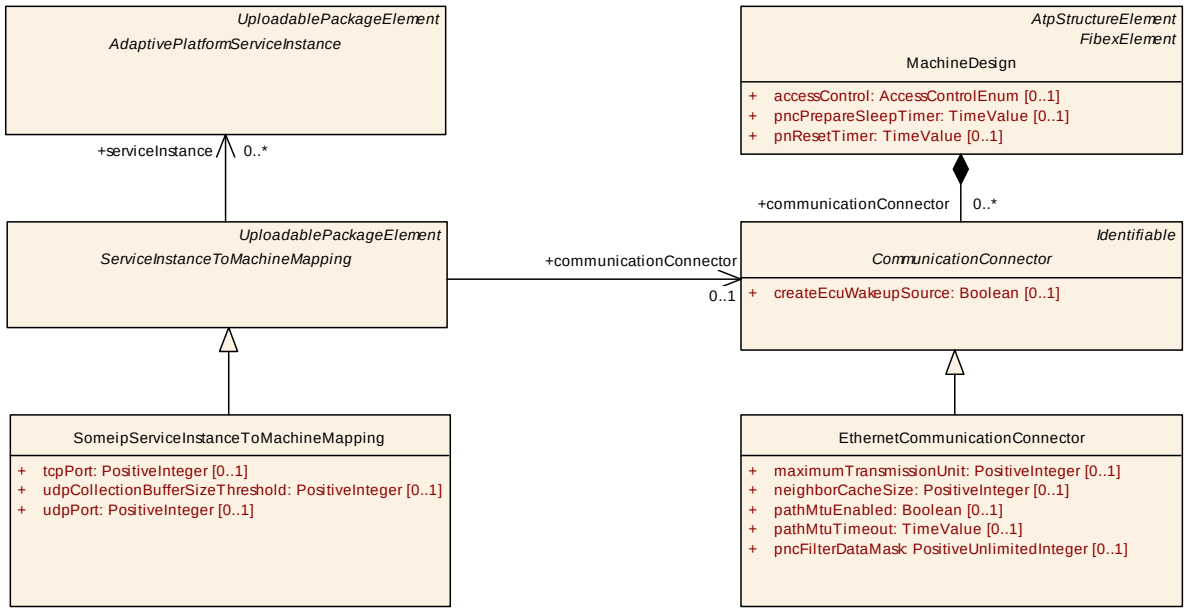


Figure 10.16: SomeipServiceInstanceToMachineMapping with TP and IP configuration

Class	<<atpVariation>> ***CommunicationCluster*** (abstract)
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology

▽

△

Class	<<atpVariation>> ***CommunicationCluster*** (abstract)			
Note	The CommunicationCluster is the main element to describe the topological connection of communicating ECUs. A cluster describes the ensemble of ECUs, which are linked by a communication medium of arbitrary topology (bus, star, ring, ...). The nodes within the cluster share the same communication protocol, which may be event-triggered, time-triggered or a combination of both. A CommunicationCluster aggregates one or more physical channels. **Tags:**vh.latestBindingTime=postBuild			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	*AbstractCanCluster*, EthernetCluster, FlexrayCluster, LinCluster, UserDefinedCluster			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
baudrate	PositiveUnlimitedInteger	0..1	attr	Channels speed in bits/s.
physical Channel	PhysicalChannel	1..*	aggr	This relationship defines which channel element belongs to which cluster. A channel shall be assigned to exactly one cluster, whereas a cluster may have one or more channels. Note: This atpSplitable property has no atp.Splitkey due to atpVariation (PropertySetPattern). **Stereotypes:** atpSplitable; atpVariation **Tags:**vh.latestBindingTime=systemDesignTime
protocolName	String	0..1	attr	The name of the protocol used.
protocolVersion	String	0..1	attr	The version of the protocol used.

Table 10.27: CommunicationCluster

Class	***CommunicationConnector*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	The connection between the referencing ECU and the referenced channel via the referenced controller. Connectors are used to describe the bus interfaces of the ECUs and to specify the sending/receiving behavior. Each CommunicationConnector has a reference to exactly one communicationController. Note: Several CommunicationConnectors can be assigned to one PhysicalChannel in the scope of one ECU Instance.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractCanCommunicationConnector*, EthernetCommunicationConnector, FlexrayCommunication Connector, LinCommunicationConnector, UserDefinedCommunicationConnector			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
createEcu WakeupSource	Boolean	0..1	attr	If this parameter is available and set to true then a channel wakeup source shall be created for the Physical Channel referencing this CommunicationConnector.

Table 10.28: CommunicationConnector

Class	**EthernetCommunicationConnector**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Ethernet specific attributes to the CommunicationConnector.			
Base	*ARObject*, *CommunicationConnector*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	EthernetCommunicationConnector			
maximum Transmission Unit	PositiveInteger	0..1	attr	This attribute specifies the maximum transmission unit in bytes.
neighborCache Size	PositiveInteger	0..1	attr	This attribute specifies the size of neighbor cache or ARP table in units of entries.
pathMtu Enabled	Boolean	0..1	attr	If enabled the IPv4/IPv6 processes incoming ICMP "Packet Too Big" messages and stores a MTU value for each destination address.
pathMtuTimeout	TimeValue	0..1	attr	If this value is >0 the IPv4/IPv6 will reset the MTU value stored for each destination after n seconds.
pncFilterData Mask	PositiveUnlimitedInteger	0..1	attr	Bit mask for Ethernet Payload used to configure the NM filter mask for the Network Management.
unicastNetwork Endpoint	NetworkEndpoint	0..1	ref	Network Endpoint that defines the IPAddress of the machine. **Tags:**atp.Status=draft

Table 10.29: EthernetCommunicationConnector

[constr_3288]{DRAFT} **IP configuration restriction for `unicastNetworkEndpoints`** ⌈A `NetworkEndpoint` that is referenced by a `EthernetCommunicationConnector` in the role `unicastNetworkEndpoint` shall have either

- `Ipv4Configuration` or
- `Ipv6Configuration`

as `networkEndpointAddress` that is defined in the unicast IP range according to the rules defined in [TPS_MANI_03005] and [TPS_MANI_03006].⌋*()*

In SOME/IP, a server that offers a `ProvidedSomeipServiceInstance` is able to send events and notification events to an IP-Multicast address.

To indicate to the client to which Multicast IP address the event messages are send the Subscribe Eventgroup Acknowledgement entry contains a reference an IPv4 Multicast Option and/or and IPv6 Multicast Option.

[TPS_MANI_03004]{DRAFT} **IPv4 Multicast event destination address** ⌈Metaclass `SomeipProvidedEventGroup` defines the multicast IPv4 address to which the `event`s and notification events of the `SomeipProvidedEventGroup` are sent to with the attribute `ipv4MulticastIpAddress`.⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03061]{DRAFT} **IPv6 Multicast event destination address** ⌈Metaclass `SomeipProvidedEventGroup` defines the multicast IPv6 address to which the `event`s and notification events of the `SomeipProvidedEventGroup` are sent to with the attribute `ipv6MulticastIpAddress`.⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03005]{DRAFT} **IPv4 Multicast address range** ⌈The IPv4 addresses reserved for multicast communication are in the range 224.0.0.0 through 239.255.255.255. Addresses between 0.0.0.0 and 223.255.255.255 are reserved for unicast communication.⌋*()*

[TPS_MANI_03006]{DRAFT} **IPv6 Multicast address range** ⌈IPv6 multicast addresses are distinguished from unicast addresses by the value of the high-order octet of the addresses: a value of 0xFF (binary 11111111) identifies an address as an address reserved for multicast communication; any other value identifies an address as a unicast address.⌋*()*

Class	***NetworkEndpointAddress*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	To build a valid network endpoint address there has to be either one MAC multicast group reference or an ipv4 configuration or an ipv6 configuration.			
Base	*ARObject*			
Subclasses	Ipv4Configuration, Ipv6Configuration, MacMulticastConfiguration			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 10.30: NetworkEndpointAddress

10.2.1.1.2 TP Configuration

The IPv4 or IPv6 Endpoint option that is referenced in the SOME/IP Offer message contains besides the IP address the transport layer protocol (e.g. UDP or TCP), and the port number of the sender.

With the `SomeipServiceInstanceToMachineMapping` the Transport Layer configuration attributes are assigned to the `ProvidedSomeipServiceInstance`.

The same element contains the Transport Layer configuration attributes for the IPv4/IPv6 Multicast Option that may be used in the SOME/IP `SubscribeEventGroupAck` message.

[TPS_MANI_03007]{DRAFT} **Udp Transport Protocol Configuration for `ProvidedSomeipServiceInstance`** ⌈The attribute `SomeipServiceInstanceToMachineMapping.udpPort` defines the Transport Protocol for a UDP communication.

This setting is used in an IPv4 or IPv6 Endpoint Option that is referenced by an `OfferService` entry.⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03008]{DRAFT} **Tcp Transport Protocol Configuration for `ProvidedSomeipServiceInstance`** ⌈The attribute `SomeipServiceInstanceToMachineMapping.tcpPort` defines the Transport Protocol for a TCP communication.

This setting is used in an IPv4 or IPv6 Endpoint Option that is referenced by an `OfferService` entry.⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03009]{DRAFT} **Tcp and Udp Transport Protocol Configuration for `ProvidedSomeipServiceInstance`** ⌈It is allowed to set `tcpPort` and `udpPort` in the same `SomeipServiceInstanceToMachineMapping`.

Such a setting shall be used to indicate that one UDP endpoint and one TCP endpoint are referenced in the `OfferService` entry. It means that the Server provides the

ProvidedSomeipServiceInstance over both Transport Protocols.⌋*(RS_MANI_00009, RS_MANI_00024)*

If a Tcp and Udp Transport Protocol Configuration is defined for a ProvidedSomeipServiceInstance as described in [TPS_MANI_03009] then the SOME/IP ServiceInterfaceDeployment settings decide which content of the ProvidedSomeipServiceInstance is transported over udp and which content is transported over tcp.

This is described in [TPS_MANI_03050] and [TPS_MANI_03051].

[TPS_MANI_03010]{DRAFT} **Udp Transport Protocol Configuration in case of IP-Multicast** ⌈The SomeipServiceInstanceToMachineMapping.eventMulticastUdpPort defines the Transport Protocol Port Number for a UDP event communication in case IP-Multicast is used.

This setting is used in an IPv4 or IPv6 Multicast Option that is referenced by a SubscribeEventGroupAck Service entry.⌋*(RS_MANI_00009, RS_MANI_00024)*

[constr_3290]{DRAFT} **Transport Protocol attributes defined for a ProvidedSomeipServiceInstance** ⌈Each SomeipServiceInstanceToMachineMapping that is defined for a ProvidedSomeipServiceInstance shall define either

- a udpPort or
- a tcpPort or
- a udpPort and a tcpPort.

⌋*()*

[TPS_MANI_03157]{DRAFT} **Enabling of data collection for UDP data transmission** ⌈The setting of the attribute SomeipServiceInstanceToMachineMapping.udpCollectionBufferSizeThreshold to a value enables the data collection for data transmission over the udpPort and unicastNetworkEndpoint defined on the EthernetCommunicationConnector that is referenced by the SomeipServiceInstanceToMachineMapping. In this case all event and method messages that are configured for data collection will be collected in the buffer until a transmission trigger arrives and the data transmission starts.⌋*(RS_MANI_00024)*

For configuration of transmission triggers please see [TPS_MANI_03158] and [TPS_MANI_03159].

10.2.1.1.3 Service Discovery Server Configuration

The multicast messages of the SOME/IP Service Discovery come with the risk of overflowing Machines with too many messages. Therefore, the Service Discovery can be configured with a suitable message sending behavior.

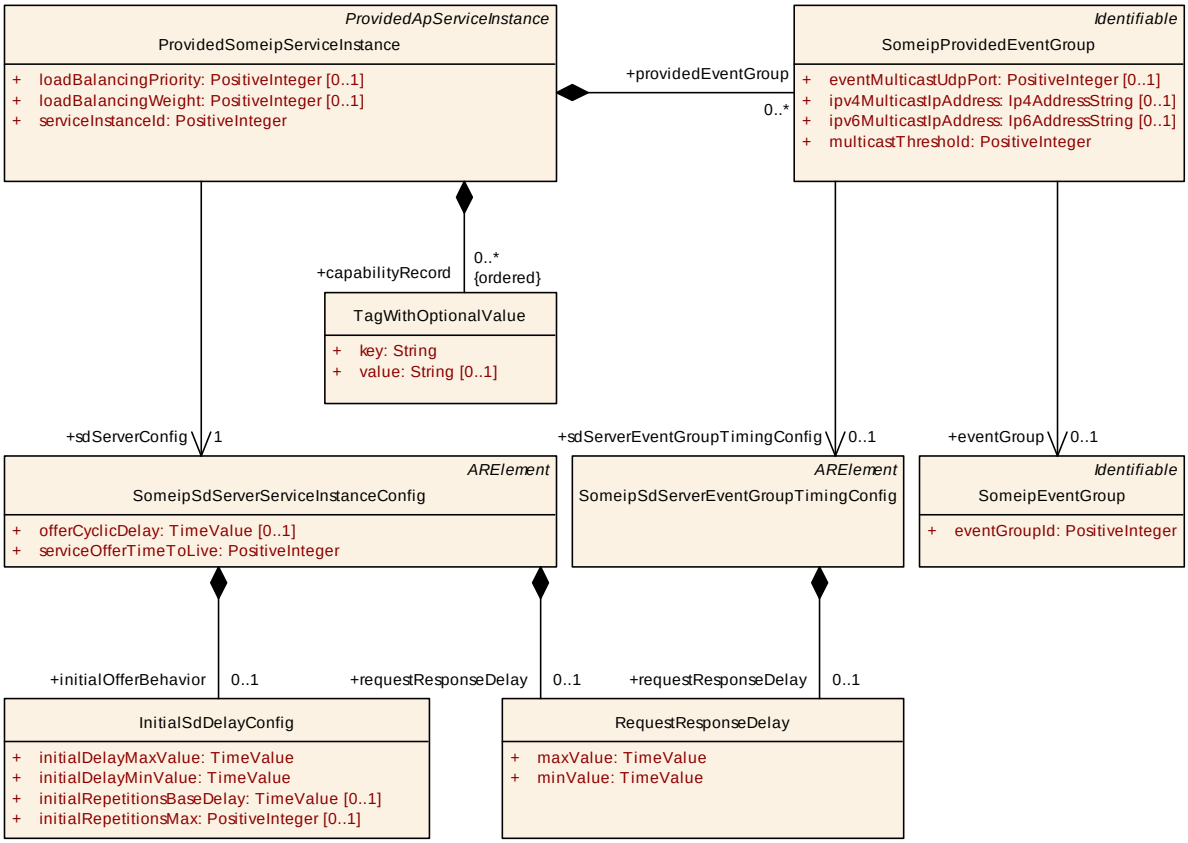


Figure 10.17: SOME/IP Service Discovery Server configuration settings

For every `ProvidedSomeipServiceInstance` on a Server different phases are existing:

- Down
- Available
  - Initial Wait Phase
  - Repetition Phase
  - Main Phase

[TPS_MANI_03011]{DRAFT} **Server Timing configuration for a `ProvidedSomeipServiceInstance`** ⌈The Server Timing is configurable with `SomeipSdServerServiceInstanceConfig` that is referenced in the role `sdServerConfig` by the `ProvidedSomeipServiceInstance` for which the Timing is valid.⌋*(RS_MANI_00024)*

The number of `SomeipSdServerServiceInstanceConfig` elements determine how many timers shall actually be used by the middleware to keep the randomized times. Via the reference `ProvidedSomeipServiceInstance.sdServerConfig` each `ProvidedSomeipServiceInstance` defines to which timer it is assigned.

If several `ProvidedSomeipServiceInstance`s share the same timer then the expiration of that timer will lead a combined sending of service discovery messages.

Note that it is possible to define several SomeipSdServerServiceInstanceConfig elements with identical timing specification values in order to request several timer handling in the middleware.

[TPS_MANI_03230]{DRAFT} **Sharing timers for ProvidedSomeipServiceInstance** ⌈If several ProvidedSomeipServiceInstances point to the same SomeipSdServerServiceInstanceConfig in the role sdServerConfig then all of these ProvidedSomeipServiceInstances will share the same timers for their timing behavior. This will lead to combining several service discovery entries in one service discovery message.⌋*(RS_MANI_00024)*

Class	**SomeipSdServerServiceInstanceConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances			
Note	Server specific settings that are relevant for the configuration of SOME/IP Service-Discovery. **Tags:**atp.recommendedPackage=SomeipSdTimingConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initialOffer Behavior	InitialSdDelayConfig	0..1	aggr	Controls offer behavior of the server.
offerCyclicDelay	TimeValue	0..1	attr	Optional attribute to define cyclic offers. Cyclic offer is active, if the delay is set (in seconds).
request ResponseDelay	RequestResponseDelay	0..1	aggr	Maximum/Minimum allowable response delay to entries received by multicast in seconds. The Service Discovery shall delay answers to entries that were transported in a multicast SOME/IP-SD message (e.g. FindService).
serviceOffer TimeToLive	PositiveInteger	1	attr	Defines the time in seconds the service offer is valid.

Table 10.31: SomeipSdServerServiceInstanceConfig

[TPS_MANI_03012]{DRAFT} **Initial Wait Phase configuration for a ProvidedSomeipServiceInstance** ⌈The Initial Wait Phase for a ProvidedSomeipServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue.

When a calculated random timer based on these min and max values expires the first OfferService entry will be sent out.⌋*(RS_MANI_00024)*

When the calculated random timer expires, the Repetition Phase will be entered.

[TPS_MANI_03013]{DRAFT} **Repetition Wait Phase configuration for a ProvidedSomeipServiceInstance** ⌈The Repetition Wait Phase for a ProvidedSomeipServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.⌋ *(RS_MANI_00024)*

If the Repetition Phase is entered the Service Discovery waits for the initialRepetitionsBaseDelay and then sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax, the Main Phase will be entered.

[TPS_MANI_03014]{DRAFT} **Main Phase configuration for a `ProvidedSomeipServiceInstance`** ⌈The Main Phase for a `ProvidedSomeipServiceInstance` is configured with the `offerCyclicDelay` attribute of `SomeipSdServerServiceInstanceConfig`.

The `OfferService` entry will be sent cyclically with an interval that is defined by the value of attribute `offerCyclicDelay`.⌋*(RS_MANI_00024)*

Class	**InitialSdDelayConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances			
Note	This element is used to configure the offer behavior of the server and the find behavior on the client.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initialDelayMaxValue	TimeValue	1	attr	Max Value in seconds to delay randomly the first offer (if aggregated in role initialOfferBehavior by SomeipSdServerServiceInstanceConfig) or the transmission of a find message (if aggregated in role initialFindBehavior by SomeipSdClientServiceInstanceConfig).
initialDelayMinValue	TimeValue	1	attr	Min Value in seconds to delay randomly the first offer (if aggregated in role initialOfferBehavior by SomeipSdServerServiceInstanceConfig) or the transmission of a find message (if aggregated in role initialFindBehavior by SomeipSdClientServiceInstanceConfig).
initialRepetitionsBaseDelay	TimeValue	0..1	attr	The base delay for offer repetitions (if aggregated in role initialOfferBehavior by SomeipSdServerServiceInstanceConfig) or find repetitions (if aggregated in role initialFindBehavior by SomeipSdClientServiceInstanceConfig). Successive find messages have an exponential back off delay.
initialRepetitionsMax	PositiveInteger	0..1	attr	Describes the maximum amount of offer repetitions (if aggregated in role initialOfferBehavior by SomeipSdServerServiceInstanceConfig) or the maximum amount of find repetitions (if aggregated in role initialFindBehavior by SomeipSdClientServiceInstanceConfig).

Table 10.32: InitialSdDelayConfig

[TPS_MANI_03015]{DRAFT} **TTL for Offer Service Entries** ⌈The lifetime of a `ProvidedSomeipServiceInstance` is configurable with the `serviceOfferTimeToLive` attribute of `SomeipSdServerServiceInstanceConfig`.

If the time that is configured by `serviceOfferTimeToLive` expires, the `ProvidedSomeipServiceInstance` is no longer offered.⌋*(RS_MANI_00024)*

[TPS_MANI_03016]{DRAFT} **Servers `RequestResponseDelay` for received `FindService` entries** ⌈The Server will delay the `OfferService` answer to a received multicast `FindService` entry by the configured `SomeipSdServerServiceInstanceConfig.requestResponseDelay`.

The actual delay will be randomly chosen between the `maxValue` and `minValue`.⌋*(RS_MANI_00024)*

Class	**RequestResponseDelay**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances			
Note	Time to wait before answering the query.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
maxValue	TimeValue	1	attr	Maximum allowable response delay to entries received by multicast in seconds.
minValue	TimeValue	1	attr	Minimum allowable response delay to entries received by multicast in seconds.

Table 10.33: RequestResponseDelay

Figure 10.18 shows an example of the different SOME/IP phases on the Server side.

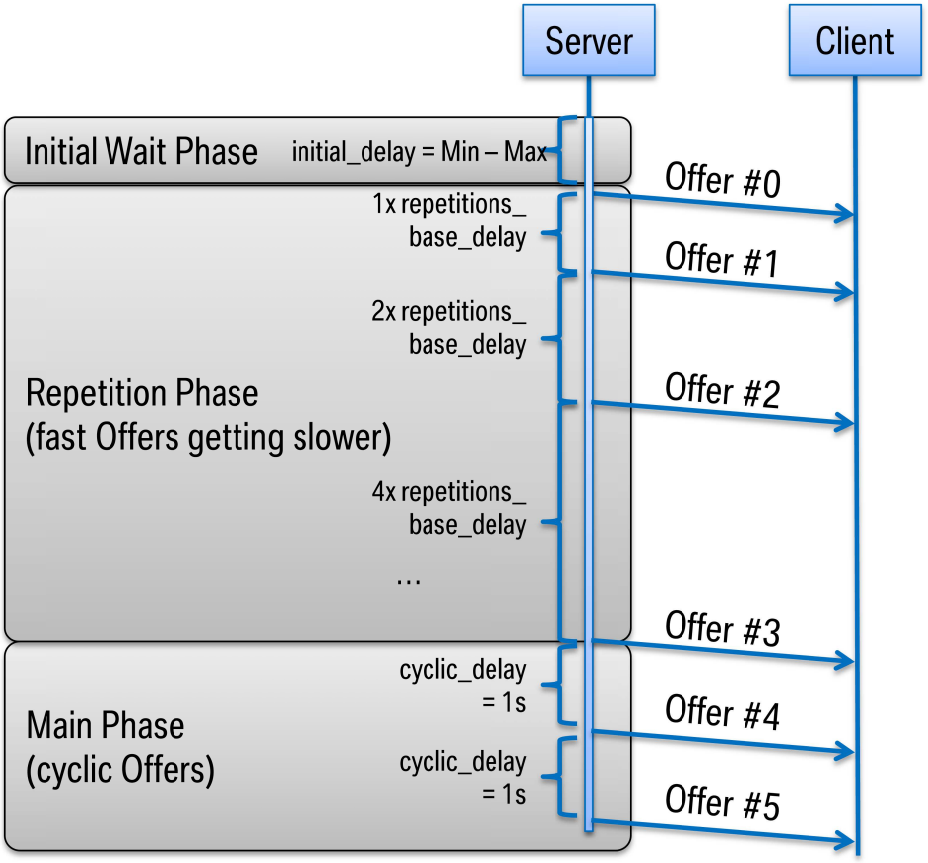


Figure 10.18: SOME/IP Server Timing example

SOME/IP allows for the specification of additional information about the `ProvidedSomeipServiceInstance` with the Capability Record that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.

[TPS_MANI_03017]{DRAFT} **Server Capability Records** ⌈A Capability Record (key/value pair) on the Server side is configurable with the `capabilityRecord` and the two attributes `key` and `value`.⌋ *(RS_MANI_00024)*

Class	**TagWithOptionalValue**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::TagWithOptionalValue			
Note	A tagged value is a combination of a tag (key) and a value that gives supplementary information that is attached to a model element. Please note that keys without a value are allowed.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
key	String	1	attr	Defines a key.
value	String	0..1	attr	Defines the corresponding value.

Table 10.34: TagWithOptionalValue

10.2.1.1.4 Provided Event Group

The `ProvidedSomeipServiceInstance` aggregates a `SomeipProvidedEventGroup` in the role `providedEventGroup` that allows to define service instance specific configuration settings for a `SomeipEventGroup`.

Class	**SomeipProvidedEventGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	The meta-class represents the ability to configure ServiceInstance related communication settings on the provided side for each EventGroup separately. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventGroup	SomeipEventGroup	0..1	ref	Reference to the SomeipEventGroup in the System Manifest for which the ServiceInstance related Event Group settings are valid. **Tags:**atp.Status=draft
eventMulticast UdpPort	PositiveInteger	0..1	attr	UdpPort configuration that is used for Event communication in the IP-Multicast case. During SOME/IP Service Discovery: Send in the SD-SubscribeEventGroupAck Message to client (answer to SD-SubscribeEventGroup). Event: This is the destination-port where the server sends the multicast event messages if the multicastThreshold is exceeded.
ipv4MulticastIp Address	Ip4AddressString	0..1	attr	Multicast IPv4 Address that is transmitted in the Event GroupSubscribeAck message.
ipv6MulticastIp Address	Ip6AddressString	0..1	attr	Multicast IPv6 Address that is transmitted in the Event GroupSubscribeAck message.
multicast Threshold	PositiveInteger	1	attr	Specifies the number of subscribed clients that trigger the server to change the transmission of events to multicast. Example: If configured to 0 only unicast will be used. If configured to 1 the first client will be already served by multicast. If configured to 2 the first client will be server ▽

▽

△

Class	SomeipProvidedEventGroup			
				△ with unicast and as soon as the 2nd client arrives both will be served by multicast. This does not influence the handling of initial events, which are served using unicast only.
sdServerEvent GroupTiming Config	SomeipSdServerEvent GroupTimingConfig	0..1	ref	Server Timing configuration settings that are EventGroup specific. **Tags:**atp.Status=draft

Table 10.35: SomeipProvidedEventGroup

[TPS_MANI_03018]{DRAFT} **Usage of `SomeipProvidedEventGroup.multicastThreshold`** ⌈The switching between IP-Unicast and IP-Multicast is guided by the server with the `SomeipProvidedEventGroup.multicastThreshold` attribute and by the number of subscribed clients to the `SomeipProvidedEventGroup`.

The Server will change the transmission of events to Multicast if the `multicastThreshold` of the corresponding `SomeipProvidedEventGroup` is reached by the number of subscribed clients. If the number of subscribed clients is smaller than the configured `multicastThreshold`, the transmission of events takes place via unicast communication.⌋ *(RS_MANI_00024)*

The following example shows the effect of the `multicastThreshold` in relation to the number of subscribed clients to the transmission of the SOME/IP event to the unicast or multicast destination address:

- If `multicastThreshold` is configured to 0, only the unicast IP address and the port will be used as destination address.
- If `multicastThreshold` is configured to 1, the first client will be served by multicast.
- If `multicastThreshold` is configured to 2, the first client will be served with unicast and as soon as the second client arrives both will be served by multicast, etc.

[TPS_MANI_03020]{DRAFT} **Servers `RequestResponseDelay` for received `SubscribeEventGroup` entries** ⌈The Server will delay the `SubscribeEventGroupAck` answer to a received `SubscribeEventGroup` message that was triggered by a multicast `ServiceOffer` by the configured `SomeipSdClientEventGroupTimingConfig.requestResponseDelay`.

The actual delay will be randomly chosen between the `maxValue` and `minValue`.⌋ *(RS_MANI_00024)*

Class	SomeipSdServerEventGroupTimingConfig			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances			
Note	EventGroup specific timing configuration settings. **Tags:**atp.recommendedPackage=SomeipSdTimingConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
request ResponseDelay	RequestResponseDelay	0..1	aggr	The Service Discovery shall delay answers to unicast messages triggered by multicast messages (e.g. Subscribe Eventgroup after Offer Service).

Table 10.36: SomeipSdServerEventGroupTimingConfig

10.2.1.1.5 `ProvidedSomeipServiceInstance` related event and method properties

[TPS_MANI_03154]{DRAFT} **`ProvidedSomeipServiceInstance` related configuration settings for `events`** ⌈The class `SomeipEventProps` that is aggregated by the `ProvidedSomeipServiceInstance` in the role `eventProps` allows for specifying `ProvidedSomeipServiceInstance` related configuration settings for `event`s that are defined in the `SomeipServiceInterfaceDeployment` referenced by the `ProvidedSomeipServiceInstance` in the role `serviceInterfaceDeployment`.⌋*(RS_MANI_00024)*

Class	SomeipEventProps			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class allows to set configuration options for an event in the provided service instance. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
collectionProps	SomeipCollectionProps	0..1	aggr	Collection of timing attributes configurable for an event that is provided by a Service Instance. **Tags:**atp.Status=draft
event	SomeipEvent Deployment	0..1	ref	Reference to the event for which the SomeipEventProps are applicable. **Tags:**atp.Status=draft

Table 10.37: SomeipEventProps

[TPS_MANI_03155]{DRAFT} **`ProvidedSomeipServiceInstance` related configuration settings for `methods`** ⌈The class `SomeipMethodProps` that is aggregated by the `ProvidedSomeipServiceInstance` in the role `methodResponseProps` allows for specifying `ProvidedSomeipServiceInstance` related configuration settings for a `method` response message. The `method` is defined in the `SomeipServiceInterfaceDeployment` referenced by the `ProvidedSomeipServiceInstance` in the role `serviceInterfaceDeployment`.⌋*(RS_MANI_00024)*

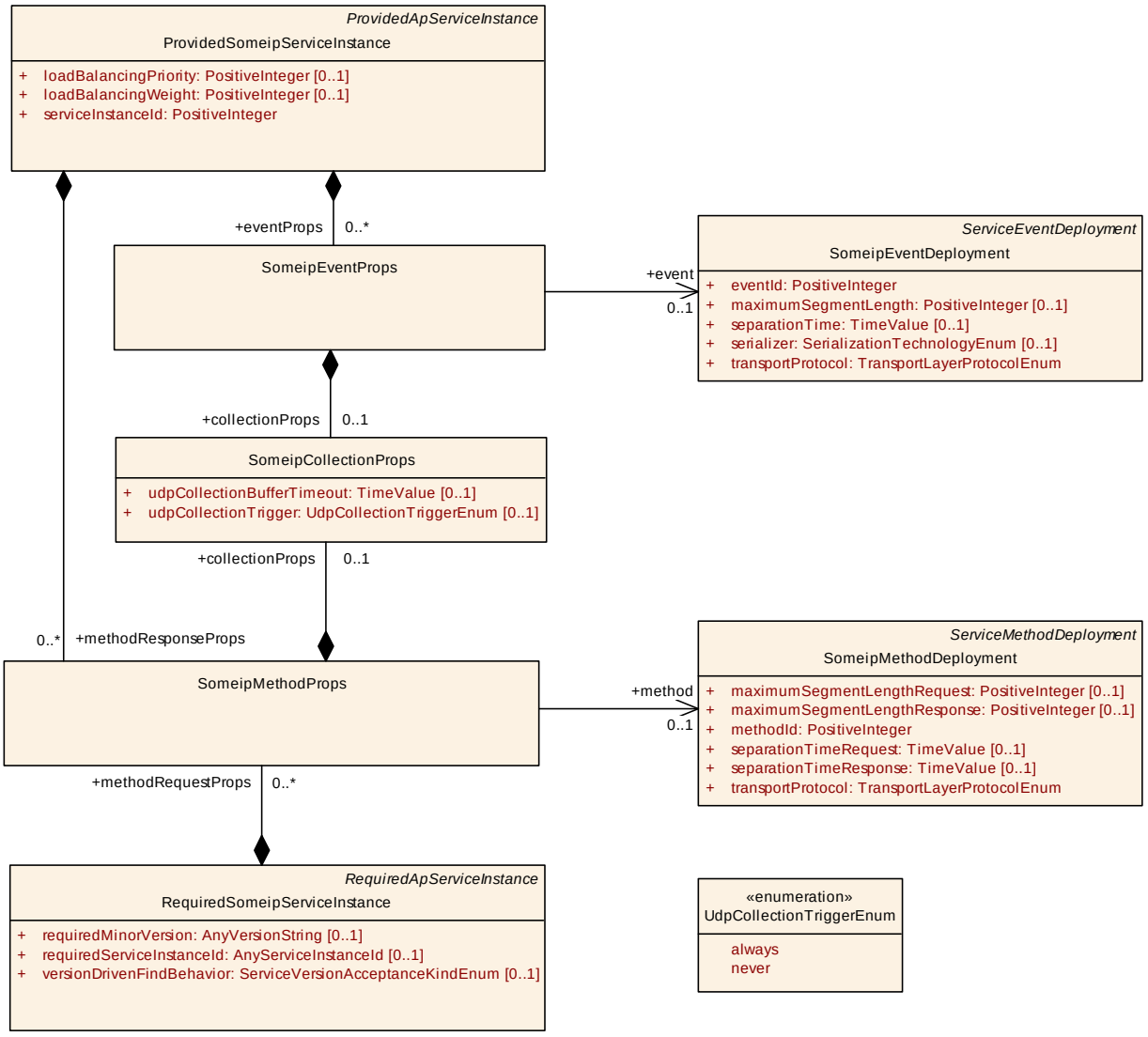


Figure 10.19: `ProvidedSomeipServiceInstance` related event and method properties

Class	**SomeipMethodProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class allows to set configuration options for a method in the service instance. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
collectionProps	SomeipCollectionProps	0..1	aggr	Collection of timing attributes configurable for a method that is provided or requested by a Service Instance. **Tags:**atp.Status=draft
method	SomeipMethod Deployment	0..1	ref	Reference to the method for which the SomeipMethod Props are applicable. **Tags:**atp.Status=draft

Table 10.38: SomeipMethodProps

Class	**SomeipCollectionProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	Collection of attributes that are configurable for an event that is provided by a ServiceInstance or for a method that is provided or requested by a ServiceInstance. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
udpCollection BufferTimeout	TimeValue	0..1	attr	Maximum time, an outgoing message (event, method call or method response) may be delayed, due to data collection.
udpCollection Trigger	UdpCollectionTrigger Enum	0..1	attr	Defines whether the ServiceInterface element (event or method) contributes to the triggering of the udp data transmission if data collection is enabled.

Table 10.39: SomeipCollectionProps

Enumeration	**UdpCollectionTriggerEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment
Note	Defines whether the ServiceInterface element (event or method) contributes to the triggering of the udp data transmission if data collection is enabled. **Tags:**atp.Status=draft
Literal	***Description***
always	ServiceInterface element will trigger the transmission of the data. **Tags:**atp.EnumerationLiteralIndex=0
never	ServiceInterface element will be buffered and will not trigger the transmission of the data. **Tags:**atp.EnumerationLiteralIndex=1

Table 10.40: UdpCollectionTriggerEnum

[TPS_MANI_03158]{DRAFT} **Configuration of a data collection on a `ProvidedServiceInstance` for transmission over udp** ⌈The attributes `udpCollectionBufferTimeout` and `udpCollectionTrigger` support the configuration of a data collection of several messages for transmission over Udp. In the `ProvidedServiceInstance` all `method` responses and `event`s for which the `udpCollectionTrigger` is set to `never` will be collected in a buffer until a trigger arrives that starts the data transmission.

The following trigger options are supported:

- a message needs to be transmitted for which the `udpCollectionTrigger` is set to `always`.
- the `udpCollectionBufferTimeout` is reached for a message.
- the buffer size defined by the attribute `udpCollectionBufferSizeThreshold` is reached.

⌋*(RS_MANI_00024)*

10.2.1.2 Required Service Instance

[TPS_MANI_03059]{DRAFT} **RequiredSomeipServiceInstance.requiredServiceInstanceId** ⌈The RequiredSomeipServiceInstance defines the requiredServiceInstanceId of a SomeipServiceInterfaceDeployment that the client searches.

The client may search for a specific requiredServiceInstanceId or for ALL requiredServiceInstanceId of the serviceInterfaceDeployment.⌋*(RS_MANI_00024)*

Class	**RequiredSomeipServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in a concrete implementation on top of SOME/IP. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInstances			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *RequiredApServiceInstance*, *Uploadable PackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
blacklisted Version	SomeipServiceVersion	*	aggr	Collection of blacklisted versions. **Tags:**atp.Status=draft
capability Record (ordered)	TagWithOptionalValue	*	aggr	A sequence of records to store arbitrary name/value pairs conveying additional information about the named service. **Tags:**atp.Status=draft
methodRequest Props	SomeipMethodProps	*	aggr	Configuration settings for individual methods that are requested by the ServiceInstance. **Tags:**atp.Status=draft
requiredEvent Group	SomeipRequiredEvent Group	*	aggr	List of EventGroups that are used by the RequiredService Instance. **Tags:**atp.Status=draft
requiredMinor Version	AnyVersionString	0..1	attr	This attribute is used to configure for which minor version of the SomeIp ServiceInterface the Service Discovery will search. Value can be set to a number that represents the Minor Version of the searched service or to ANY.
requiredService InstanceId	AnyServiceInstanceId	0..1	attr	This attribute represents the ability to describe the required service instance ID. **Tags:**atp.Status=draft
sdClientConfig	SomeipSdClientService InstanceConfig	1	ref	Client specific configuration settings relevant for the SOME/IP service discovery. **Tags:**atp.Status=draft
versionDriven FindBehavior	ServiceVersion AcceptanceKindEnum	0..1	attr	Defines the service discovery find behavior.

Table 10.41: RequiredSomeipServiceInstance

Class	**SomeipServiceVersion**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances			
Note	This meta-class represents the ability to describe a version of a SOME/IP Service. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
majorVersion	PositiveInteger	0..1	attr	Major Version of the ServiceInterface. **Tags:**xml.sequenceOffset=10
minorVersion	PositiveInteger	1	attr	Minor Version of the ServiceInterface. **Tags:**xml.sequenceOffset=20

Table 10.42: SomeipServiceVersion

Enumeration	**ServiceVersionAcceptanceKindEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances
Note	Defined the possible acceptance kinds for required service instances. **Tags:**atp.Status=draft
Literal	***Description***
exactOrAnyMinor Version	Search for ANY or specific minor version service instance and select either ALL returned service instances (in case of ANY) or exactly the specific minor version service instances defined in required MinorVersion. **Tags:**atp.EnumerationLiteralIndex=0
minimumMinor Version	Search for ANY minor version service instance and select only those service instances which have an equal or greater minor version than given in requiredMinorVersion. **Tags:**atp.EnumerationLiteralIndex=1

Table 10.43: ServiceVersionAcceptanceKindEnum

[TPS_MANI_03021]{DRAFT} **Requirements on the searched minor version from the client's point of view** ⌈The meta-class `RequiredSomeipServiceInstance` is able to make further specifications regarding the version of the service from the client's point of view.

For this purpose, the attribute `RequiredSomeipServiceInstance.requiredMinorVersion` exists and provides the ability to define the required minor version (`SomeipServiceVersion.minorVersion`).⌋*(RS_MANI_00024)*

Please note that the major version that the client searches for is already defined by the `SomeipServiceVersion.majorVersion` in the `SomeipServiceInterfaceDeployment`. It is therefore not possible to search for ANY major version, so the client looks always for a specific major version.

The minor version that may be defined by `SomeipServiceVersion.minorVersion` in the `SomeipServiceInterfaceDeployment` is irrelevant for the client and the service search and shall be ignored.

[TPS_MANI_03619]{DRAFT} **SOME/IP Service search for `requiredMinorVersion`** ⌈A `RequiredSomeipServiceInstance` is searching for a SOME/IP Service Instance `requiredMinorVersion`:

- in case versionDrivenFindBehavior = exactOrAnyMinorVersion: Service minor version that matches the value set in requiredMinorVersion or *ANY* minor version of the Service Instance in case the requiredMinorVersion is set to *ANY*
- in case versionDrivenFindBehavior = minimumMinorVersion: Service minor version that matches at least the value set in requiredMinorVersion or is higher

⌋()

[constr_3561]{DRAFT} **minimumMinorVersion and RequiredSomeipServiceInstance.requiredMinorVersion value** ⌈The RequiredSomeipServiceInstance.requiredMinorVersion shall not have the value *ANY* if versionDrivenFindBehavior = minimumMinorVersion.⌋()

[TPS_MANI_03618]{DRAFT} **Usage of RequiredSomeipServiceInstance.blacklistedVersion** ⌈A service connection of a RequiredSomeipServiceInstance to a ProvidedSomeipServiceInstance is not considered for service discovery if the SomeipServiceVersion.minorVersion of the enclosing SomeipServiceInterfaceDeployment that is referenced by the ProvidedSomeipServiceInstance exists in the collection of SomeipServiceVersions aggregated at the RequiredSomeipServiceInstance in the role blacklistedVersion.⌋(*RS_MANI_00066*)

A typical scenario for using a blacklist may be: For a certain RequiredSomeipServiceInstance a certain compatible provider service version inside a system may not work which may have been identified after the design phase. In order to keep the system running this certain provider version won't be considered in the service search if it has been blacklisted. Therefore, the RequiredSomeipServiceInstance may connect only to ProvidedSomeipServiceInstances that fulfill the search criteria and are not blacklisted.

[constr_3558]{DRAFT} **RequiredSomeipServiceInstance.blacklistedVersion is restricted to the usage of minorVersion** ⌈The majorVersion attribute shall not be used in the SomeipServiceVersion that is aggregated by the RequiredSomeipServiceInstance in the role blacklistedVersion.⌋()

[constr_5115]{DRAFT} **Search for a specific SOME/IP ServiceInstance and for all SOME/IP ServiceInstances over the same RPortPrototype** ⌈A RequiredSomeipServiceInstance that configures the search for a specific ServiceInstance on SOME/IP (with concrete requiredServiceInstanceId) and a RequiredSomeipServiceInstance that configures the search for ALL ServiceInstances on SOME/IP (with requiredServiceInstanceId = ALL) that are mapped using ServiceInstanceToMachineMapping to the same EthernetCommunicationConnector (and therefore are searching for SOME/IP ServiceInstances on the same VLAN) are not allowed to be mapped by ServiceInstanceToPortPrototypeMappings to the same RPortPrototype.⌋()

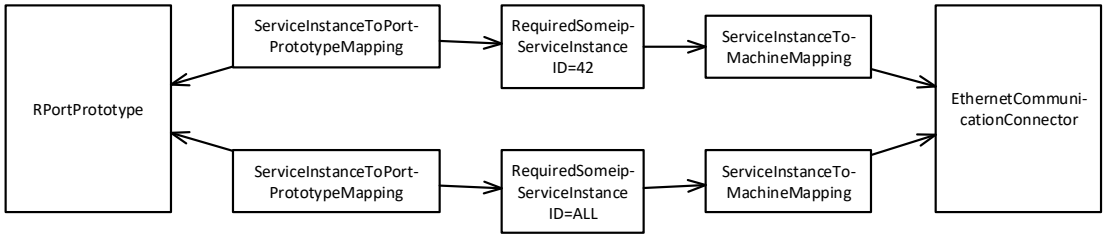


Figure 10.20: Depiction of not-allowed scenario

Please note that the depicted scenario in Figure 10.20 is covered by [constr_5115].

In other words the middleware is allowed to search for a concrete SOME/IP `ServiceInstance` on one VLAN and for ALL SOME/IP `ServiceInstance`s on a different VLAN via a single `RPortPrototype`. But the middleware is only able to search for either ALL SOME/IP `ServiceInstance`s or for one concrete SOME/IP `ServiceInstance` on the same VLAN via a single `RPortPrototype`.

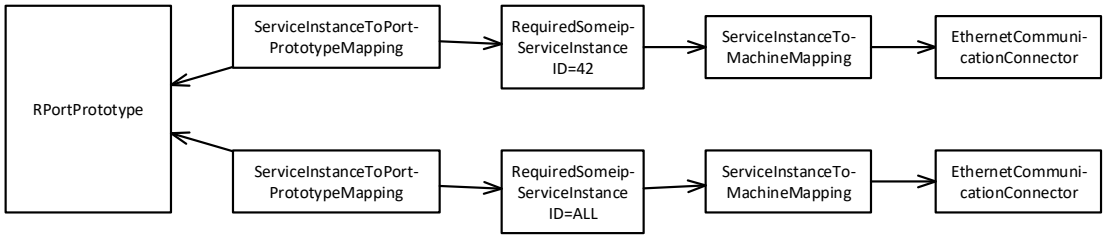


Figure 10.21: Depiction of allowed scenario

Please note that the depicted scenario in Figure 10.21 is covered by [constr_5115].

10.2.1.2.1 IP Configuration

In SOME/IP, the `SubscribeEventGroup` entry references IPv4 or IPv6 Endpoint options to indicate to the server where the client wants to receive the events of the `SomeipEventGroup`. Such an Endpoint contains the IP address of the client.

[TPS_MANI_03022]{DRAFT} **Context of `RequiredSomeipServiceInstance`** ⌈A `RequiredSomeipServiceInstance` can be mapped to a `CommunicationConnector` of a `MachineDesign` with the `SomeipServiceInstanceToMachineMapping`.

With this mapping an assignment of the `RequiredSomeipServiceInstance` to a unicast IP Address is established since the `EthernetCommunicationConnector` refers to a `NetworkEndpoint` in the role `unicastNetworkEndpoint`. The `unicastNetworkEndpoint` defines the local IP address of the client.⌋*(RS_MANI_00009, RS_MANI_00024)*

10.2.1.2.2 TP Configuration

The IPv4 or IPv6 Endpoint option that is referenced in the SOME/IP `SubscribeEventGroup` message contains besides the IP address the transport layer protocol (e.g. UDP or TCP), and the port number of the client.

With the `SomeipServiceInstanceToMachineMapping` the Transport Layer configuration attributes are assigned to the `RequiredSomeipServiceInstance`.

The Transport Layer (TCP/UDP) configuration attributes for the `SubscribeEventGroup` entry are directly available in the `SomeipServiceInstanceToMachineMapping` element.

The `SomeipServiceInstanceToMachineMapping` defines also the source-port where the client sends the method call messages to the server and the destination-port where the client receives the method responses from the server.

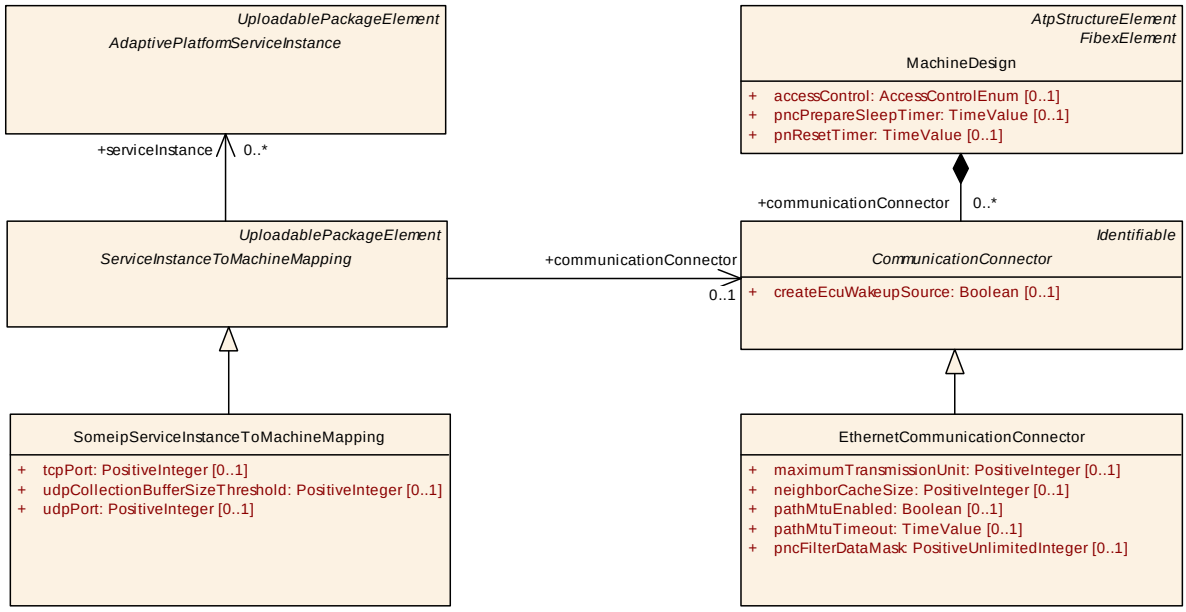


Figure 10.22: `SomeipServiceInstanceToMachineMapping` with TP and IP configuration

[TPS_MANI_03023]{DRAFT} **Udp Transport Protocol Configuration for `RequiredSomeipServiceInstance`** ⌈The `SomeipServiceInstanceToMachineMapping.udpPort` defines the Transport Protocol for a UDP communication in case that the server provides `ServiceInterface` content over UDP and the client wants to use it.⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03024]{DRAFT} **Tcp Transport Protocol Configuration for `RequiredSomeipServiceInstance`** ⌈The `SomeipServiceInstanceToMachineMapping.tcpPort` defines the Transport Protocol for a TCP communication in case that the server provides `ServiceInterface` content over TCP and the client wants to use it.⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03049]{DRAFT} **Tcp and Udp Transport Protocol Configuration for RequiredSomeipServiceInstance** ⌈It is allowed to set tcpPort and udpPort in the same SomeipServiceInstanceToMachineMapping. Such a setting shall be used in case that the server provides ServiceInterface content over Udp and Tcp and the client wants to use it.⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03237]{DRAFT} **Transport Protocol attributes defined for a RequiredSomeipServiceInstance** ⌈Each SomeipServiceInstanceToMachineMapping that is defined for a RequiredSomeipServiceInstance is allowed to have:

- a configured udpPort or
- a configured tcpPort or
- a configured udpPort and a tcpPort or
- no configured udpPort and tcpPort.

⌋*(RS_MANI_00009, RS_MANI_00024)*

A RequiredSomeipServiceInstance that is mapped by a SomeipServiceInstanceToMachineMapping that does not contain a udpPort and tcpPort is allowed to receive events over IP Multicast only. In this case it is not required for a Client to have a unicast socket prepared if the server will always use the multicast transport. In such a case the SubscribeEventGroup SOME/IP SD message that is sent from the ServiceConsumer to the ServiceProvider will not contain any Unicast Endpoint options.

In addition the corresponding ServiceInterface is not allowed to have any Methods defined since the Request/Response communication pattern is restricted to IP Unicast only. An additional prerequisite for such a setup is that the ProvidedSomeipServiceInstance is configured for IP multicast transmission only. In other words all SomeipProvidedEventGroups need to be set multicastThreshold = 1.

[constr_5161]{DRAFT} **RequiredSomeipServiceInstance that is mapped by a SomeipServiceInstanceToMachineMapping without a configured tcpPort and udpPort** ⌈A RequiredSomeipServiceInstance that is mapped to a EthernetCommunicationConnector by a SomeipServiceInstanceToMachineMapping that does not have neither a udpPort nor a tcpPort is not allowed to reference a SomeipServiceInterfaceDeployment that includes SomeipMethodDeployments (directly or indirectly via ServiceFieldDeployment).⌋*()*

If a Tcp and Udp Transport Protocol Configuration is defined for a RequiredSomeipServiceInstance as described in [TPS_MANI_03049] then the SOME/IP ServiceInterfaceDeployment settings decide which content of the ProvidedSomeipServiceInstance is transported over udp and which content is transported over tcp. This is described in [TPS_MANI_03050] and [TPS_MANI_03051].

10.2.1.2.3 Service Discovery Client Configuration

Service Discovery phases on the Client side allow minimizing the number of Service Discovery messages and allow a fast synchronization upon ECU start.

For every RequiredSomeipServiceInstance on a Client different phases are existing:

- Down
- Requested
  - Initial Wait Phase
  - Repetition Phase
  - Main Phase

[TPS_MANI_03025]{DRAFT} **Client Timing configuration for a RequiredSomeipServiceInstance** ⌈The Client Timing is configurable with SomeipSdClientServiceInstanceConfig that is referenced in the role sdClientConfig by the RequiredSomeipServiceInstance for which the Timing is valid.⌋*(RS_MANI_00024)*

The number of SomeipSdClientServiceInstanceConfig elements determine how many timers shall actually be used by the middleware to keep the randomized times. Via the reference RequiredSomeipServiceInstance.sdClientConfig each RequiredSomeipServiceInstance defines to which timer it is assigned.

Note that it is possible to define several SomeipSdClientServiceInstanceConfig elements with identical timing specification values in order to request several timer handling in the middleware.

If several RequiredSomeipServiceInstance share the same timer then the expiration of that timer will lead a combined sending of service discovery messages.

[TPS_MANI_03231]{DRAFT} **Sharing timers for RequiredSomeipServiceInstance** ⌈If several RequiredSomeipServiceInstances point to the same SomeipSdClientServiceInstanceConfig in the role sdClientConfig then all of these RequiredSomeipServiceInstances will share the same timers for their timing behavior. This will lead to combining several service discovery entries in one service discovery message.⌋*(RS_MANI_00024)*

Class	**SomeipSdClientServiceInstanceConfig**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances
Note	Client specific settings that are relevant for the configuration of SOME/IP Service-Discovery. **Tags:**atp.recommendedPackage=SomeipSdTimingConfigs
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*

▽

△

Class	**SomeipSdClientServiceInstanceConfig**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initialFind Behavior	InitialSdDelayConfig	0..1	aggr	Controls initial find behavior of clients.
serviceFind TimeToLive	PositiveInteger	1	attr	This attribute represents the ability to define the time in seconds the service find is valid.

Table 10.44: SomeipSdClientServiceInstanceConfig

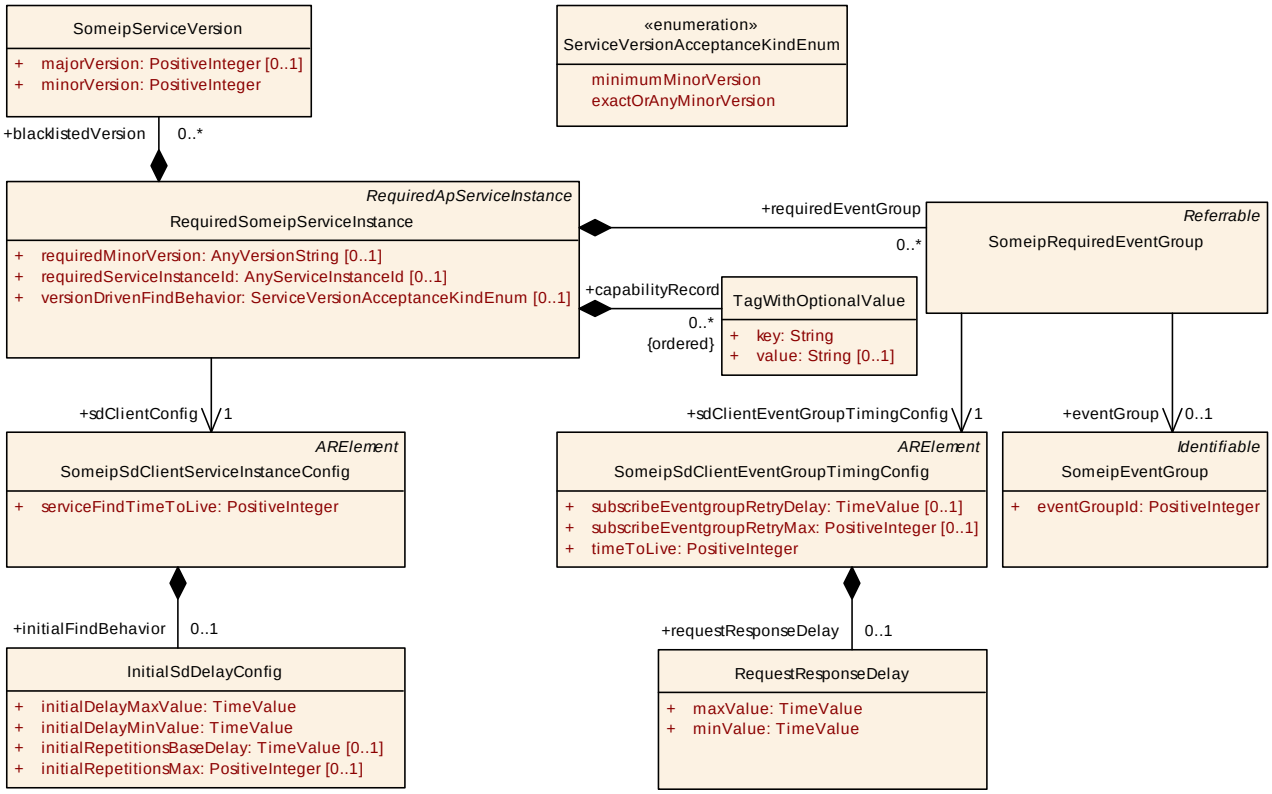


Figure 10.23: SOME/IP Service Discovery Client configuration settings

[TPS_MANI_03026]{DRAFT} **Initial Wait Phase configuration for a `RequiredSomeipServiceInstance`** ⌈The Initial Wait Phase for a `RequiredSomeipServiceInstance` is configured with the `initialFindBehavior` and the two attributes `initialDelayMinValue` and `initialDelayMaxValue`.

If a calculated random timer based on these min and max values expires the first `FindService` entry will be sent out.⌋*(RS_MANI_00024)*

When the calculated random timer expires and no `OfferService` is received the Repetition Phase will be entered.

[TPS_MANI_03027]{DRAFT} **Repetition Wait Phase configuration for a `RequiredSomeipServiceInstance`** ⌈The Repetition Wait Phase for a `RequiredSomeipServiceInstance` is configured with the `initialFindBehavior` and the two attributes `initialRepetitionsMax` and `initialRepetitionsBaseDelay`.⌋ *(RS_MANI_00024)*

If the Repetition Phase is entered, the Service Discovery waits the `initialRepetitionsBaseDelay` and sends an `FindService` entry.

If the amount of sent `FindService` entries reaches `initialRepetitionsMax` and no `OfferService` is received the Main Phase will be entered. In the Main Phase no further `FindService` entries are sent by the client.

[TPS_MANI_03028]{DRAFT} **TTL for Find Service Entries** ⌈The lifetime of a `RequiredSomeipServiceInstance` is configurable with the `serviceFindTimeToLive` attribute of `SomeipSdClientServiceInstanceConfig`.

If the time that is configured by `serviceFindTimeToLive` expires, the `FindService` entry shall be considered not existing.⌋*(RS_MANI_00024)*

Figure 10.24 shows an example of the different SOME/IP phases on the Client side.

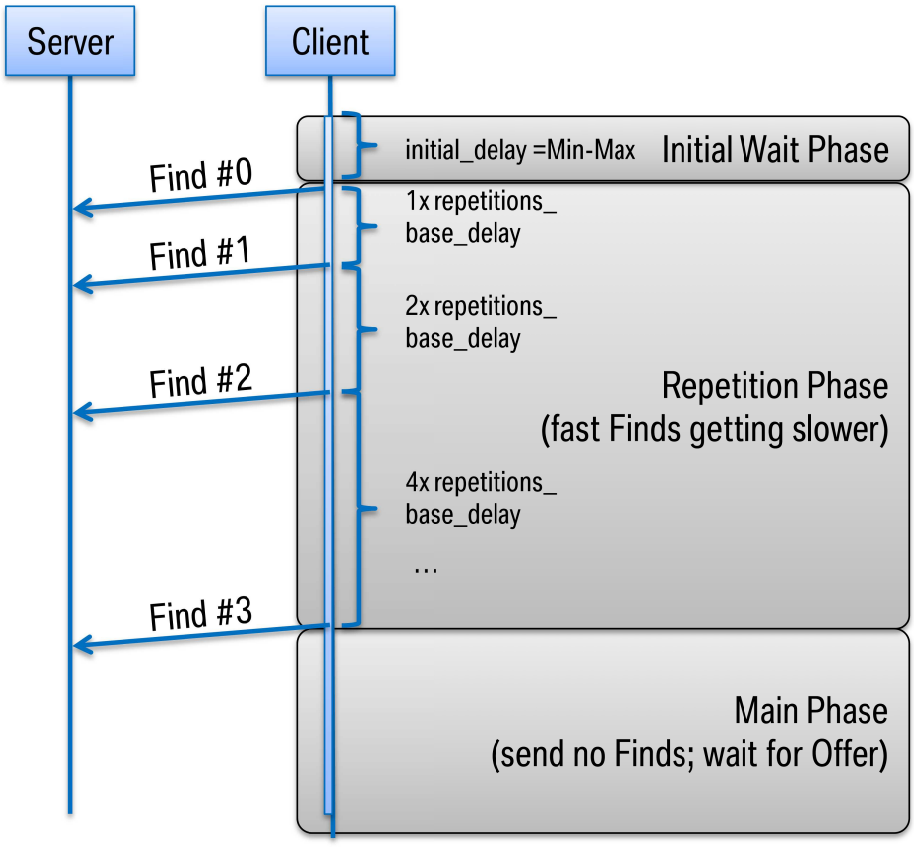


Figure 10.24: SOME/IP Client Timing example

SOME/IP allows for specifying additional information about the `RequiredSomeipServiceInstance` with the Capability Record that allows to transport arbitrary configuration strings (key/value pairs).

This allows to encode additional information like the name of a service or its configuration.

[TPS_MANI_03029]{DRAFT} **Client Capability Records** ⌈A Capability Record (key/-value pair) on the Client side is configurable with the `capabilityRecord` and the two attributes `key` and `value`.⌋*(RS_MANI_00024)*

10.2.1.2.4 Required Event Group

The `RequiredSomeipServiceInstance` aggregates a `SomeipRequiredEvent-Group` in the role `requiredEventGroup` that allows to define service instance specific configuration settings for a `SomeipEventGroup`.

Class	**SomeipRequiredEventGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	The meta-class represents the ability to configure ServiceInstance related communication settings on the required side for each EventGroup separately. **Tags:**atp.Status=draft			
Base	*ARObject*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventGroup	SomeipEventGroup	0..1	ref	Reference to the SomeipEventGroup in the System Manifest for which the ServiceInstance related Event Group settings are valid. **Tags:**atp.Status=draft
sdClientEvent GroupTiming Config	SomeipSdClientEvent GroupTimingConfig	1	ref	Client Timing configuration settings that are EventGroup specific. **Tags:**atp.Status=draft

Table 10.45: SomeipRequiredEventGroup

Class	**SomeipSdClientEventGroupTimingConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances			
Note	This meta-class is used to specify configuration related to service discovery in the context of an event group on SOME/IP. **Tags:**atp.recommendedPackage=SomeipSdTimingConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
request ResponseDelay	RequestResponseDelay	0..1	aggr	The Service Discovery shall delay answers to unicast messages triggered by multicast messages (e.g. Subscribe Eventgroup after Offer Service).
subscribe Eventgroup RetryDelay	TimeValue	0..1	attr	This attribute defines the interval in seconds to re-trigger a subscription to a Eventgroup, if a retry to subscribe to a Eventgroup is configured (subscribeEventgroupRetryMax > 0).
subscribe Eventgroup RetryMax	PositiveInteger	0..1	attr	This attribute define the maximum counts of retries to subscribe to an Eventgroup. If the value is set to 0 no retry shall be done. If the value is set to 255 the retry shall be done as along as the Eventgroup is requested and no SubscribeEventGroupAck was received.

▽

△

Class	SomeipSdClientEventGroupTimingConfig			
timeToLive	PositiveInteger	1	attr	Defines the time in seconds the subscription of this event is expected by the client. this value is sent from the client to the server in the SD-subscribeEvent message.

Table 10.46: SomeipSdClientEventGroupTimingConfig

[TPS_MANI_03030]{DRAFT} **`SomeipSdClientEventGroupTimingConfig`.`timeToLive` for `SubscribeEventGroup` Entries** ⌈The lifetime of an event subscription is configurable with the `timeToLive` attribute of `SomeipSdClientEventGroupTimingConfig`.

If the time that is configured by `timeToLive` expires, the event subscription is canceled.⌋*(RS_MANI_00024)*

[TPS_MANI_03031]{DRAFT} **Clients `RequestResponseDelay` for received `ServiceOffer` entries** ⌈The Client will delay the `SubscribeEventGroup` answer to a received `ServiceOffer` message by the configured `SomeipSdClientEventGroupTimingConfig.requestResponseDelay`.

The actual delay will be randomly chosen between the `maxValue` and `minValue`.⌋*(RS_MANI_00024)*

10.2.1.2.5 `RequiredSomeipServiceInstance` related method call properties

[TPS_MANI_03156]{DRAFT} **`RequiredSomeipServiceInstance` related configuration settings for `method`s** ⌈The class `SomeipMethodProps` that is aggregated by the `RequiredSomeipServiceInstance` in the role `methodRequestProps` allows specifying `RequiredSomeipServiceInstance` related configuration settings for a `method` request message. The `method` is defined in the `SomeipServiceInterfaceDeployment` referenced by the `RequiredSomeipServiceInstance` in the role `serviceInterfaceDeployment`.⌋*(RS_MANI_00024)*

[TPS_MANI_03159]{DRAFT} **Configuration of a data collection on a `RequiredSomeipServiceInstance` for transmission over udp** ⌈The attributes `udpCollectionBufferTimeout` and `udpCollectionTrigger` support the configuration of a data collection of several messages for transmission over udp. In the `RequiredSomeipServiceInstance` all `method` requests for which the `udpCollectionTrigger` is set to `never` will be collected in a buffer until a trigger arrives that starts the data transmission.

The following trigger options are supported:

- a message needs to be transmitted for which the `udpCollectionTrigger` is set to `always`.
- the `udpCollectionBufferTimeout` is reached for a message.

- the buffer size defined by the attribute udpCollectionBufferSizeThreshold is reached.

⌋(*RS_MANI_00024*)

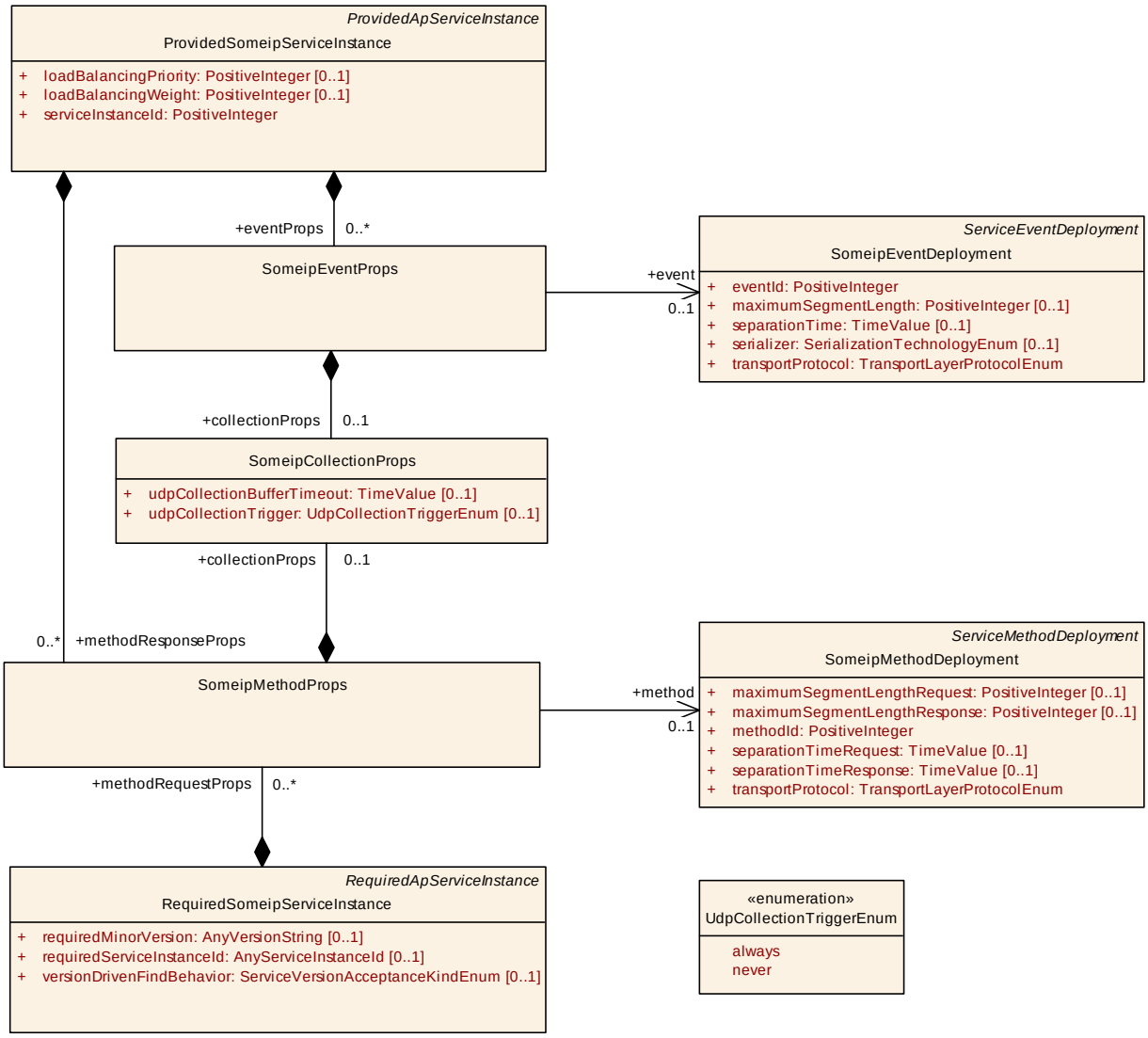


Figure 10.25: RequiredSomeipServiceInstance related event and method properties

10.2.2 DDS Service Instance Deployment

In the case of DDS used as the transport layer the derived meta-classes are DdsProvidedServiceInstance or DdsRequiredServiceInstance. These meta-classes also carry attributes that apply for the service discovery on DDS.

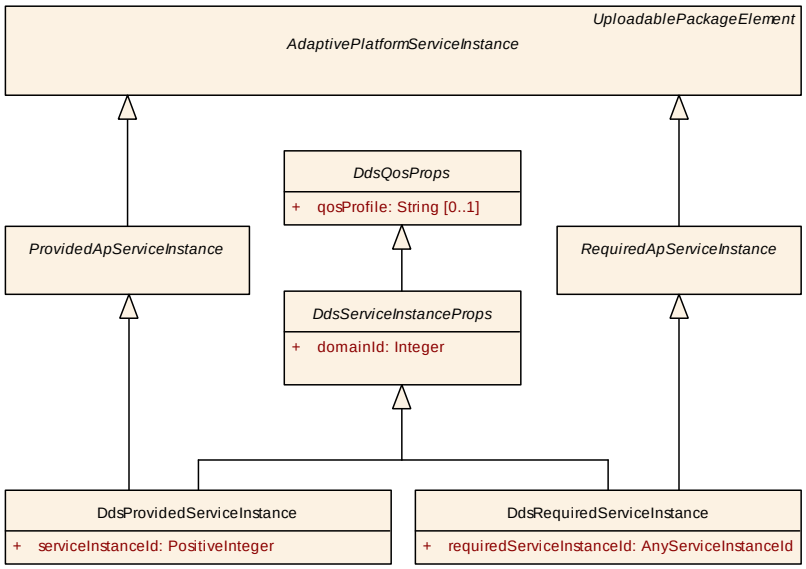


Figure 10.26: Dds Service Instances

Class	***DdsQosProps*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	QoS configuration properties for the DDS entities associated with an event, method, or field provided by or requested from a Service Instance using DDS as the underlying network binding. **Tags:**atp.Status=draft			
Base	*ARObject*			
Subclasses	DdsEventQosProps, DdsFieldQosProps, *DdsServiceInstanceProps*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
qosProfile	String	0..1	attr	Identifies a group of QoS Policies that apply to the DDS entities associated with the event, method, field, or the service instance. **Tags:**atp.Status=draft

Table 10.47: DdsQosProps

Class	***DdsServiceInstanceProps*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	Common configuration properties for the DDS entities provided by or requested from a Service Instance using DDS as the underlying network binding. **Tags:**atp.Status=draft			
Base	*ARObject*, *DdsQosProps*			
Subclasses	DdsProvidedServiceInstance, DdsRequiredServiceInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
domainId	Integer	1	attr	This attribute identifies the DDS Domain the Service Instance shall join. **Tags:**atp.Status=draft

Table 10.48: DdsServiceInstanceProps

10.2.2.1 Provided DDS Service Instance

[TPS_MANI_03527]{DRAFT} **Definition of `DdsProvidedServiceInstance`** ⌈The `DdsProvidedServiceInstance` configures the Service to join a DDS Domain with the `domainId` attribute, and to instantiate the underlying DDS entities according to a QoS Profile with the `qosProfile` attribute. Moreover, it assigns an Instance ID to the Service for deployment with the `serviceInstanceId` attribute.⌋*(RS_MANI_00038)*

[constr_3528]{DRAFT} **Value range of `domainId`** ⌈The value of `domainId` at `DdsProvidedServiceInstance` and `domainId` at `DdsRequiredServiceInstance` shall be in the range of a signed 32-bit integer.⌋*()*

[constr_3529]{DRAFT} **Value range of `serviceInstanceId`** ⌈The value of `serviceInstanceId` shall be in the range of 0..65535.⌋*()*

[constr_3541]{DRAFT} **`qosProfile` mandatory for `DdsProvidedServiceInstance`** ⌈The attribute `qosProfile` shall be defined for every `DdsProvidedServiceInstance`.⌋*()*

[constr_3564]{DRAFT} **Consistency between DDS Service Interface Deployment and Provided DDS Service Instance** ⌈Transport attributes `DdsServiceInterfaceDeployment.transportProtocol` and `DdsEventDeployment.transportProtocol` shall be consistent with DDS profiles generated and selected by the `DdsQosProps` component of `DdsProvidedServiceInstance`, `DdsFieldQosProps`, and `DdsEventQosProps`.⌋*()*

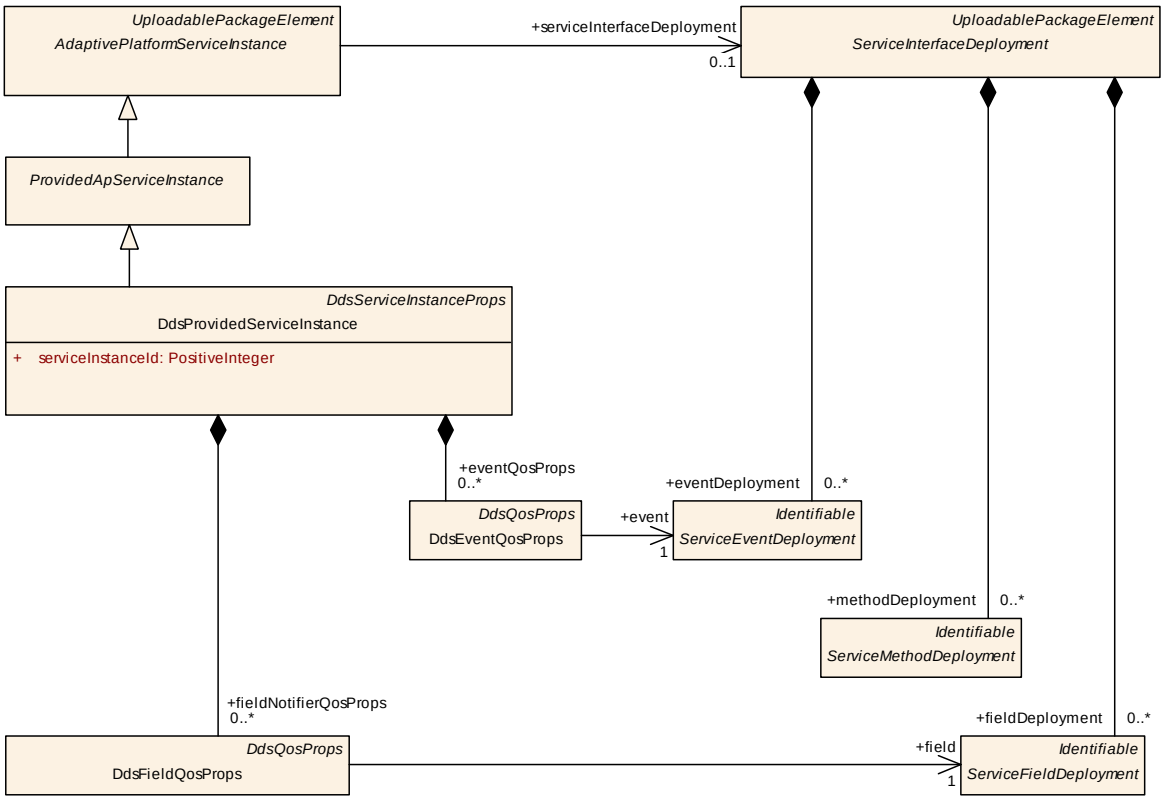


Figure 10.27: Provided Dds Service Instances

Class	DdsProvidedServiceInstance			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in a concrete implementation on top of DDS. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInstances			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *DdsQosProps*, *DdsServiceInstanceProps*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *ProvidedApServiceInstance*, *Referrable*, *UploadablePackageElement*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
eventQosProps	DdsEventQosProps	*	aggr	List of configuration properties for the Events that are provided by the Service Instance. **Tags:**atp.Status=draft
fieldNotifierQosProps	DdsFieldQosProps	*	aggr	List of configuration properties for Field notifiers that are provided by the Service Instance. **Tags:**atp.Status=draft
serviceInstanceId	PositiveInteger	1	attr	Identification number that is used by DDS to identify DomainParticipants associated with an instance of the service. **Tags:**atp.Status=draft

Table 10.49: DdsProvidedServiceInstance

[TPS_MANI_03528]{DRAFT} **Definition of `DdsProvidedServiceInstance.eventQosProps`** ⌈The `DdsProvidedServiceInstance.eventQosProps` configures the DDS entities associated with the `event` according to a QoS Profile specified with the `qosProfile` attribute.⌋*(RS_MANI_00038)*

[TPS_MANI_03531]{DRAFT} **`qosProfile` of `DdsProvidedServiceInstance.eventQosProps` is optional** ⌈The attribute `qosProfile` of `DdsProvidedServiceInstance.eventQosProps` is optional; if `qosProfile` is not defined, the underlying DDS entities shall be configured according to the `qosProfile` attribute of the parent `DdsProvidedServiceInstance`.⌋*(RS_MANI_00038)*

Class	DdsEventQosProps			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	Configuration properties of the Event using DDS as the underlying network binding. **Tags:**atp.Status=draft			
Base	*ARObject*, *DdsQosProps*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
event	ServiceEventDeployment	1	ref	Reference to an event that is provided. **Tags:**atp.Status=draft

Table 10.50: DdsEventQosProps

[TPS_MANI_03561]{DRAFT} **Definition of `DdsProvidedServiceInstance.fieldNotifierQosProps`** ⌈The `DdsProvidedServiceInstance.fieldNotifierQosProps` configures the DDS entities associated with the `field` according to a QoS Profile specified with the `qosProfile` attribute.⌋*(RS_MANI_00038)*

[TPS_MANI_03562]{DRAFT} **qosProfile of DdsProvidedServiceInstance.fieldNotifierQosProps is optional** ⌈The attribute qosProfile of DdsProvidedServiceInstance.fieldNotifierQosProps is optional; if qosProfile is not defined, the underlying DDS entities shall be configured according to the qosProfile attribute of the parent DdsProvidedServiceInstance.⌋*(RS_MANI_00038)*

Class	**DdsFieldQosProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	Configuration properties of the Field interaction when using DDS as the underlying network binding. **Tags:**atp.Status=draft			
Base	*ARObject*, *DdsQosProps*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
field	ServiceField Deployment	1	ref	Reference to the field. **Tags:**atp.Status=draft

Table 10.51: DdsFieldQosProps

10.2.2.2 Required DDS Service Instance

[TPS_MANI_03529]{DRAFT} **Definition of DdsRequiredServiceInstance** ⌈The DdsRequiredServiceInstance configures the Client to join a DDS Domain with the domainId attribute, and to instantiate the underlying DDS entities according to a QoS Profile with the qosProfile attribute. Optionally, the requiredServiceInstanceId attribute allows a Client to search for a specific Instance ID of the serviceInterface.⌋*(RS_MANI_00038)*

[constr_3542]{DRAFT} **qosProfile mandatory for DdsRequiredServiceInstance** ⌈The attribute qosProfile shall be defined for every DdsRequiredServiceInstance.⌋*()*

[constr_3565]{DRAFT} **Consistency between DDS Service Interface Deployment and Required DDS Service Instance** ⌈Transport attributes DdsServiceInterfaceDeployment.transportProtocol and DdsEventDeployment.transportProtocol shall be consistent with DDS profiles generated and selected by the DdsQosProps component of DdsRequiredServiceInstance, DdsFieldQosProps, and DdsEventQosProps.⌋*()*

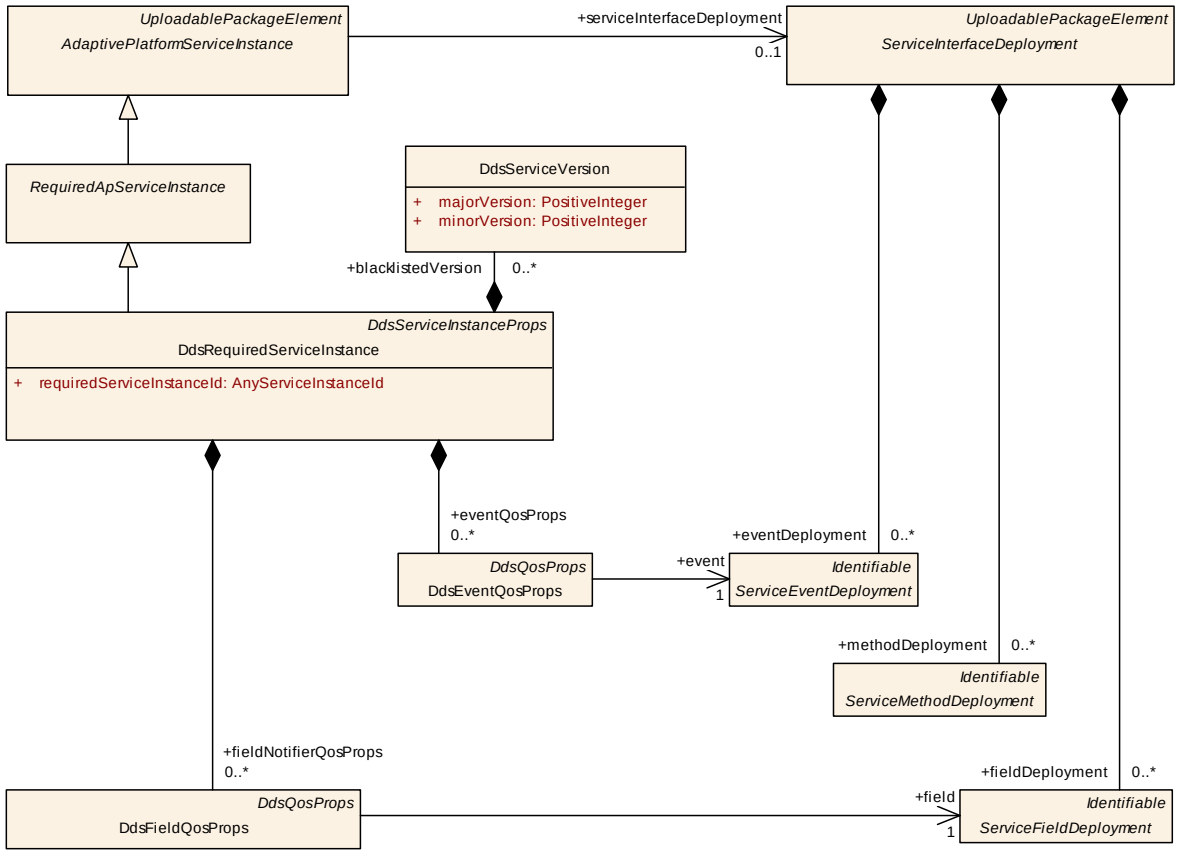


Figure 10.28: Required Dds Service Instances

Class	**DdsRequiredServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in a concrete implementation on top of DDS. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInstances			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *DdsQosProps*, *DdsServiceInstanceProps*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *RequiredApServiceInstance*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
blacklisted Version	DdsServiceVersion	*	aggr	Collection of blacklisted versions. **Tags:**atp.Status=draft
eventQosProps	DdsEventQosProps	*	aggr	List of configuration properties for the Events that are required by the Service Instance. **Tags:**atp.Status=draft
fieldNotifierQos Props	DdsFieldQosProps	*	aggr	List of configuration properties for Field notifiers that are required by the Service Instance. **Tags:**atp.Status=draft

△

Class	**DdsRequiredServiceInstance**			
requiredService InstanceId	AnyServiceInstanceId	1	attr	This attribute represents the ability to describe the required service instance ID. **Tags:**atp.Status=draft

Table 10.52: DdsRequiredServiceInstance

[TPS_MANI_03530]{DRAFT} **Definition of `DdsRequiredServiceInstance.eventQosProps`** ⌈The `DdsRequiredServiceInstance.eventQosProps` configures the DDS entities responsible for subscribing to an `event` according to a QoS Profile specified with the `qosProfile` attribute.⌋*(RS_MANI_00038)*

[TPS_MANI_03532]{DRAFT} **`qosProfile` of `DdsRequiredServiceInstance.eventQosProps` is optional** ⌈The attribute `qosProfile` of `DdsRequiredServiceInstance.eventQosProps` is optional; if `qosProfile` is not defined, the underlying DDS entities shall be configured according to the `qosProfile` attribute of the parent `DdsRequiredServiceInstance`.⌋*(RS_MANI_00038)*

[TPS_MANI_03567]{DRAFT} **Definition of `DdsRequiredServiceInstance.fieldNotifierQosProps`** ⌈The `DdsRequiredServiceInstance.fieldNotifierQosProps` configures the DDS entities associated with the `field` according to a QoS Profile specified with the `qosProfile` attribute.⌋*(RS_MANI_00038)*

[TPS_MANI_03568]{DRAFT} **`qosProfile` of `DdsRequiredServiceInstance.fieldNotifierQosProps` is optional** ⌈The attribute `qosProfile` of `DdsRequiredServiceInstance.fieldNotifierQosProps` is optional; if `qosProfile` is not defined, the underlying DDS entities shall be configured according to the `qosProfile` attribute of the parent `DdsRequiredServiceInstance`.⌋*(RS_MANI_00038)*

10.2.2.3 DDS Service Instance to Machine mapping

The `DdsServiceInstanceToMachineMapping` defines on which network / VLAN the DDS communication shall be deployed.

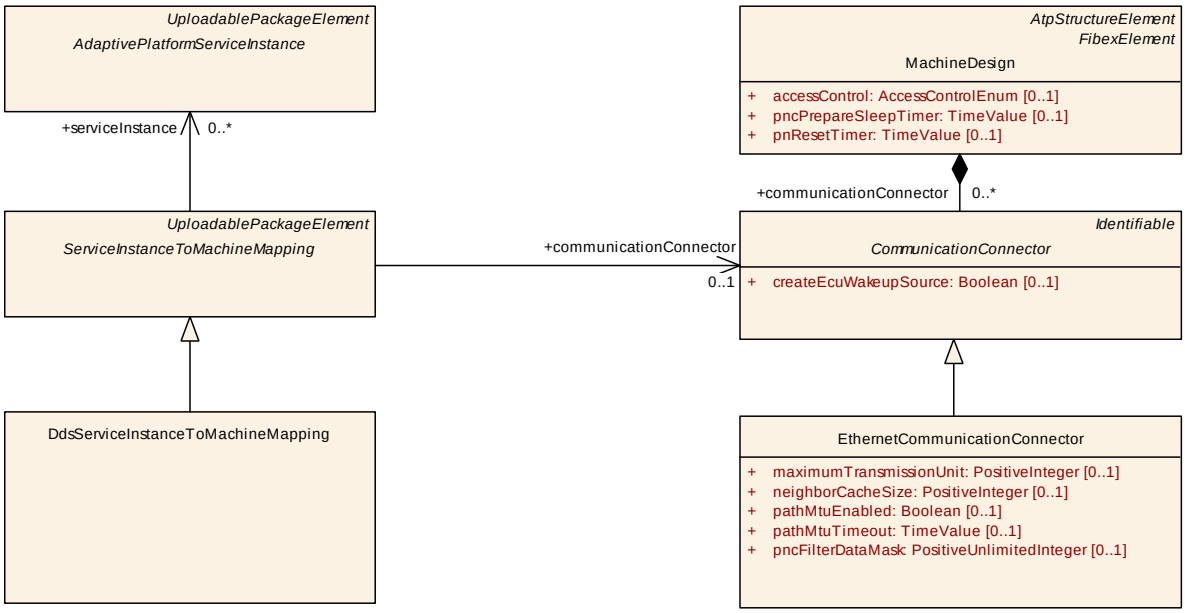


Figure 10.29: Dds Service Instance to Machine mapping

[TPS_MANI_03533]{DRAFT} **DdsServiceInstanceToMachineMapping** ⌈The DdsServiceInstanceToMachineMapping defines for a specific serviceInstance (either DdsProvidedServiceInstance or DdsRequiredServiceInstance) on which network the communication shall be done using the reference communicationConnector to CommunicationConnector.⌋*(RS_MANI_00038)*

Class	**DdsServiceInstanceToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	This meta-class allows to map DdsServiceInstances to a CommunicationConnector of a Machine. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInstanceToMachineMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *ServiceInstanceToMachineMapping*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 10.53: DdsServiceInstanceToMachineMapping

10.2.3 User Defined Service Instance Deployment

[TPS_MANI_03032]{DRAFT} **Description of middleware technologies not standardized by AUTOSAR** ⌈The elements ProvidedUserDefinedServiceInstance and RequiredUserDefinedServiceInstance can be used to describe alternative middleware technologies that are not standardized by AUTOSAR.⌋*(RS_MANI_00014)*

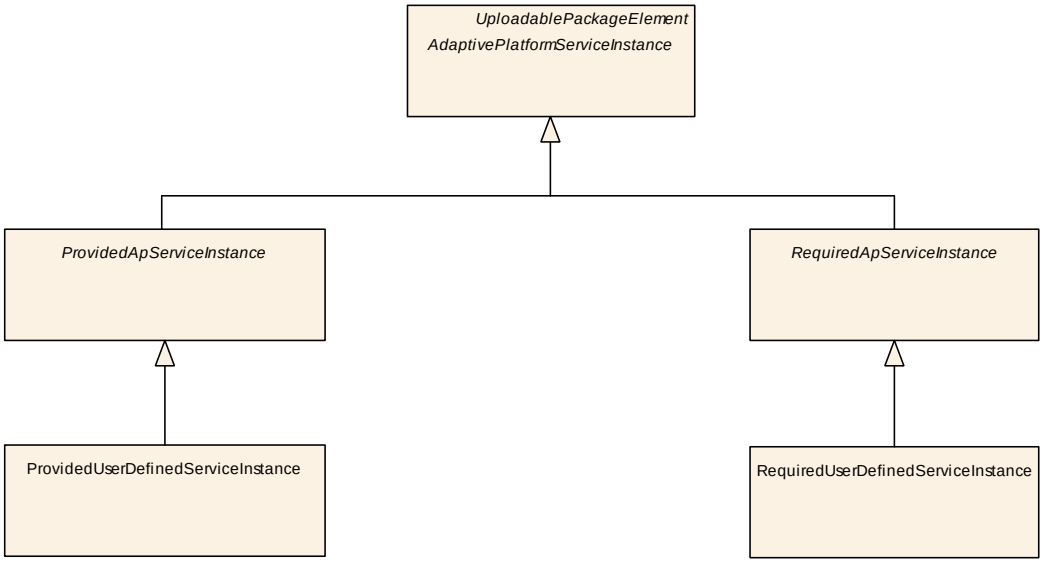


Figure 10.30: User Defined Service Instance Deployment

Class	**ProvidedUserDefinedServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in a concrete implementation that is not standardized by AUTOSAR. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInstances			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *ProvidedApServiceInstance*, *Referrable*, *Uploadable PackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 10.54: ProvidedUserDefinedServiceInstance

Class	**RequiredUserDefinedServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in a concrete implementation that is not standardized by AUTOSAR. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInstances			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *RequiredApServiceInstance*, *Uploadable PackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 10.55: RequiredUserDefinedServiceInstance

Please note that both elements `ProvidedUserDefinedServiceInstance` and `RequiredUserDefinedServiceInstance` are `Identifiable` and therefore are able to describe special data (`sdg`) which is not represented by the standard model.

10.3 EndToEndProtection

AUTOSAR supports the protection of event s, method s, Field notifier s, Field get methods and Field set methods with E2E Profiles that are defined in the E2E Communication Protection Library [28].

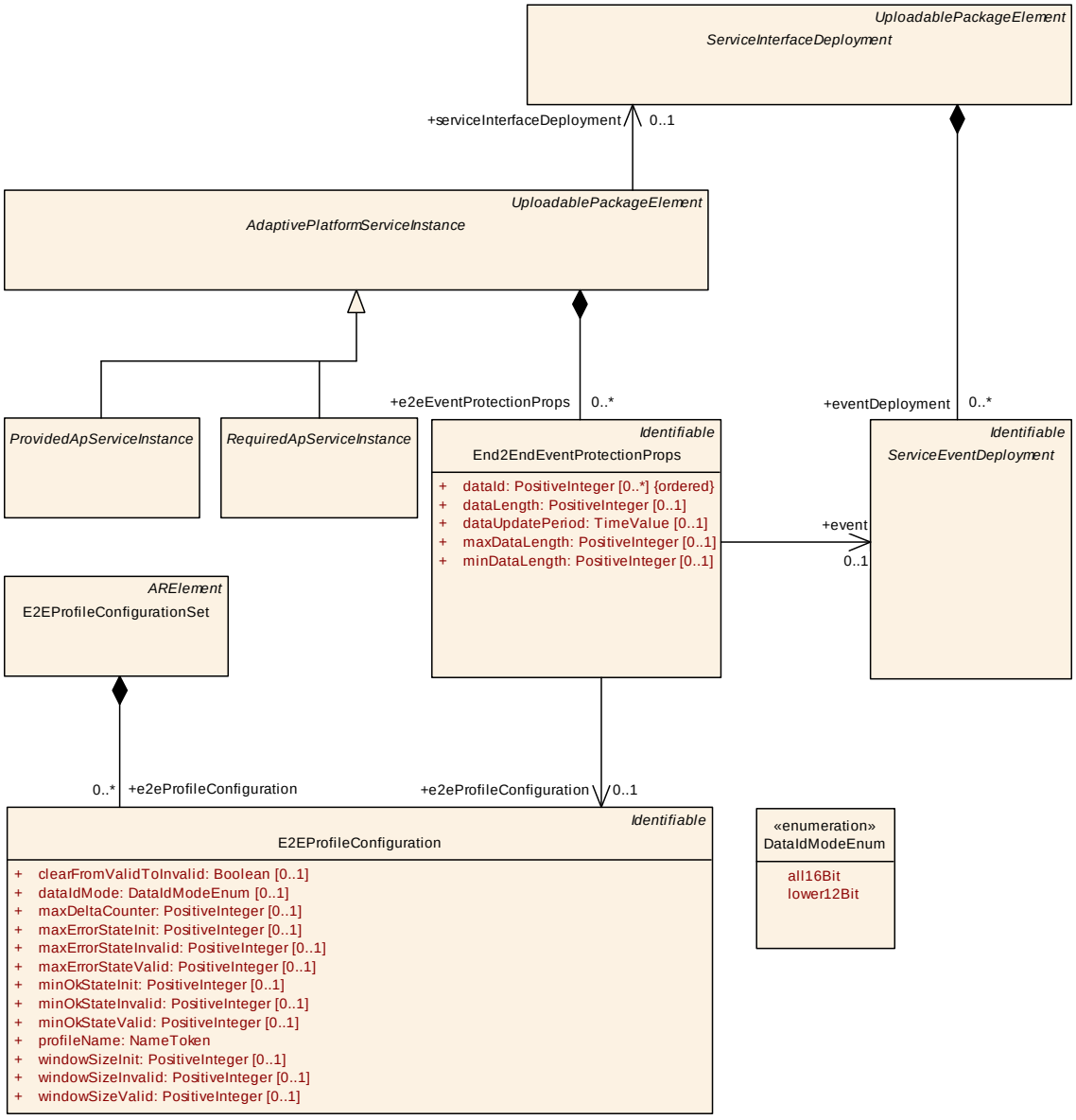


Figure 10.31: E2E EventProtection

[TPS_MANI_03127]{DRAFT} **Usage of End2EndEventProtectionProps** ⌈The End2EndEventProtectionProps element is used to define event specific E2E configuration settings in the context of an AdaptivePlatformServiceInstance.⌋ *(RS_MANI_00028)*

Please note that the E2E protection of a field notifier is possible with the End2EndEventProtectionProps.event reference since each specific ServiceFieldDeployment element aggregates a ServiceEventDeployment in the role

notifier. If such an aggregated ServiceEventDeployment is referenced with the End2EndEventProtectionProps.event reference the E2E protection settings are valid for the notifier that is embedded by the ServiceFieldDeployment.

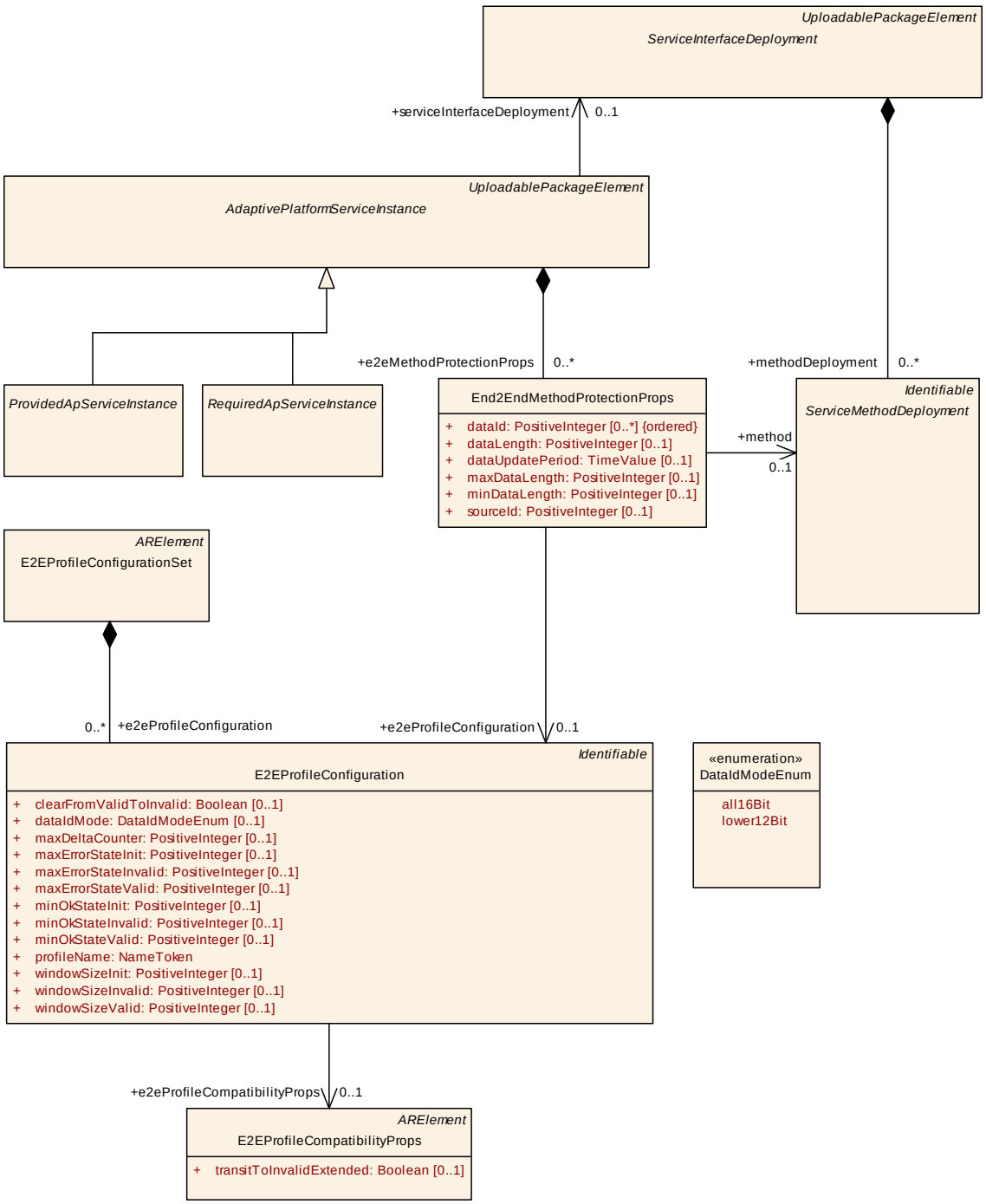


Figure 10.32: E2E MethodProtection

Since the End2EndEventProtectionProps element is aggregated by the abstract AdaptivePlatformServiceInstance it can be used to describe the End-to-End

protection on specific derived classes like ProvidedSomeipServiceInstance or RequiredSomeipServiceInstance that fit the underlying middleware.

With this approach it is possible to define different End-to-End protection settings for different used transport layer mechanisms in case of Multi-Binding.

[TPS_MANI_03228]{DRAFT} **Usage of End2EndMethodProtectionProps** ⌈The End2EndMethodProtectionProps element is used to define method specific E2E configuration settings in the context of an AdaptivePlatformServiceInstance.⌋ *(RS_MANI_00028)*

Please note that the E2E protection of field get and set methods is possible with the End2EndMethodProtectionProps.method reference since each specific ServiceFieldDeployment element is allowed to aggregate a ServiceMethodDeployment in the role get and/or set.

If such an aggregated ServiceMethodDeployment is referenced with the End2EndMethodProtectionProps.method reference the E2E protection settings are valid for the get or set method that is embedded by the ServiceFieldDeployment.

[TPS_MANI_03129]{DRAFT} **E2E profile** ⌈The E2E profile is defined by E2EProfileConfiguration.profileName.⌋ *(RS_MANI_00028)*

[TPS_MANI_03130]{DRAFT} **Standardized E2EProfileConfiguration.profileName values** ⌈The E2EProfileConfiguration.profileName that is referenced by an End2EndEventProtectionProps or by an End2EndMethodProtectionProps can have the following values that are standardized by AUTOSAR: PROFILE_04, PROFILE_05, PROFILE_06, PROFILE_07, PROFILE_08, PROFILE_11, PROFILE_22, PROFILE_04m, PROFILE_07m, and PROFILE_44.⌋ *(RS_MANI_00028)*

[TPS_MANI_03131]{DRAFT} **Non-Standardized E2EProfileConfiguration.profileName values** ⌈The values for the profileName of E2EProfileConfiguration mentioned in [TPS_MANI_03130] are standardized and reserved for being used in the way the AUTOSAR standard foresees. PROFILE_01 and PROFILE_02 are also reserved by AUTOSAR but excluded for usage in Adaptive AUTOSAR. In addition, it is positively possible to use other than the standardized values for the profileName.⌋ *(RS_MANI_00028)*

[TPS_MANI_03128]{DRAFT} **Usage of same End2EndEventProtectionProps.dataId in case of Multi-Binding** ⌈In case of Multi-Binding, i.e. if different AdaptivePlatformServiceInstances exist that are mapped by ServiceInstanceToPortPrototypeMapping to the same PortPrototype, the different AdaptivePlatformServiceInstances may contain the same dataId for the same event.⌋ *(RS_MANI_00028)*

In other words if a PortPrototype contains two transport layer bindings, e.g. a ProvidedSomeipServiceInstance and a ProvidedUserDefinedServiceInstance representing an IPC communication then an event is allowed to be protected

with the same dataId in both AdaptivePlatformServiceInstances because the two AdaptivePlatformServiceInstances effectively represent the identical piece of data.

[TPS_MANI_03229]{DRAFT} **Usage of same End2EndMethodProtectionProps.dataId in case of Multi-Binding** ⌈In case of Multi-Binding, i.e. if different AdaptivePlatformServiceInstances exist that are mapped by ServiceInstanceToPortPrototypeMapping to the same PortPrototype, the different AdaptivePlatformServiceInstances may contain the same dataId for the same method.⌋*(RS_MANI_00028)*

In other words if a PortPrototype contains two transport layer bindings, e.g. a ProvidedSomeipServiceInstance and a ProvidedUserDefinedServiceInstance representing an IPC communication then a method is allowed to be protected with the same dataId in both AdaptivePlatformServiceInstances.

[TPS_MANI_03252]{DRAFT} **Usage of same End2EndMethodProtectionProps.sourceId in case of Multi-Binding** ⌈In case of Multi-Binding, i.e. if different AdaptivePlatformServiceInstances exist that are mapped by ServiceInstanceToPortPrototypeMapping to the same PortPrototype, the different AdaptivePlatformServiceInstances may contain the same sourceId (for the same and even for a different method).⌋*(RS_MANI_00028)*

In other words if a PortPrototype contains two transport layer bindings, e.g. a ProvidedSomeipServiceInstance and a ProvidedUserDefinedServiceInstance representing an IPC communication then a single as well as different methods are allowed to be protected with the same sourceId in both AdaptivePlatformServiceInstances.

Class	**End2EndEventProtectionProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::E2E			
Note	This element allows to protect an event or a field notifier with an E2E profile. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataId (ordered)	PositiveInteger	*	attr	This represents a unique numerical identifier for the referenced event or field notifier that is included in the CRC calculation. Note: ID is used for protection against masquerading. The details concerning the maximum number of values (this information is specific for each E2E profile) applicable for this attribute are controlled by a semantic constraint that depends on the category of the EndToEnd Protection.
dataLength	PositiveInteger	0..1	attr	Length of payload including E2E header in bits.
dataUpdate Period	TimeValue	0..1	attr	This attribute describes the period in which the applications are assumed to process E2E-protected messages. The middleware does not use this attribute at all.

▽

△

Class	**End2EndEventProtectionProps**			
e2eProfile Configuration	E2EProfileConfiguration	0..1	ref	Reference to E2E profile configuration settings that are valid to protect the referenced event or field notifier. **Tags:**atp.Status=draft
event	ServiceEvent Deployment	0..1	ref	Reference to an event that is protected by the E2E profile. **Tags:**atp.Status=draft
maxDataLength	PositiveInteger	0..1	attr	Maximum length of payload including E2E header in bits.
minDataLength	PositiveInteger	0..1	attr	Minimum length of payload including E2E header in bits.

Table 10.56: End2EndEventProtectionProps

Class	**End2EndMethodProtectionProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::E2E			
Note	This element allows to protect a method, a field setter or a field getter with an E2E profile. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataId (ordered)	PositiveInteger	*	attr	This represents a numerical identifier that is included in the CRC calculation. This dataId is used for call and response. Note: ID is used for protection against masquerading. The details concerning the maximum number of values (this information is specific for each E2E profile) applicable for this attribute are controlled by a semantic constraint that depends on the category of the EndToEnd Protection.
dataLength	PositiveInteger	0..1	attr	Length of payload including E2E header in bits.
dataUpdate Period	TimeValue	0..1	attr	This attribute describes the period in which the applications are assumed to process E2E-protected messages. The middleware does not use this attribute at all.
e2eProfile Configuration	E2EProfileConfiguration	0..1	ref	Reference to E2E profile configuration settings that are valid to protect the referenced method, field getter or field setter. **Tags:**atp.Status=draft
maxDataLength	PositiveInteger	0..1	attr	Maximum length of payload including E2E header in bits.
method	ServiceMethod Deployment	0..1	ref	Reference to a method, a field getter or a field setter that is protected by the E2E profile. **Tags:**atp.Status=draft
minDataLength	PositiveInteger	0..1	attr	Minimum length of payload including E2E header in bits.
sourceId	PositiveInteger	0..1	attr	This represents a unique numerical identifier identifying the source of a certain transmission. In case of C/S communication, this ID uniquely identifies the client. Note: ID is used for protection against masquerading. The details concerning the maximum number of values (this information is specific for each E2E profile) applicable for this attribute are controlled by a semantic constraint that depends on the category of the EndToEnd Protection.

Table 10.57: End2EndMethodProtectionProps

Class	E2EProfileConfigurationSet			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::E2E			
Note	This meta-class represents the ability to aggregate a collection of E2EProfileConfigurations. **Tags:** atp.Status=draft atp.recommendedPackage=E2EProfileConfigurationSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
e2eProfile Configuration	E2EProfileConfiguration	*	aggr	This represents the collection of E2EProfileConfigurations aggregated at the E2EProfileConfigurationSet. **Tags:**atp.Status=draft

Table 10.58: E2EProfileConfigurationSet

Class	E2EProfileConfiguration			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::E2E			
Note	This element holds E2E profile specific configuration settings. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
clearFromValid ToInvalid	Boolean	0..1	attr	Clear monitoring window on transition from state Valid to state Invalid.
dataIdMode	DataIdModeEnum	0..1	attr	This attribute describes the inclusion mode that is used to include the implicit Data ID in the one-byte CRC.
e2eProfile Compatibility Props	E2EProfileCompatibility Props	0..1	ref	Reference to additional settings for the E2E state machine. **Tags:**atp.Status=draft
maxDelta Counter	PositiveInteger	0..1	attr	Maximum allowed difference between two counter values of two consecutively received valid messages. For example, if the receiver gets data with counter 1 and Max DeltaCounter is 3, then at the next reception the receiver can accept Counters with values 2, 3 or 4.
maxErrorState Init	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_INIT.
maxErrorState Invalid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_INVALID.
maxErrorState Valid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_VALID.
minOkStateInit	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INIT.
minOkState Invalid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INVALID.
minOkState Valid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_VALID.

△

Class	**E2EProfileConfiguration**			
profileName	NameToken	1	attr	Definition of the E2E profile.
windowSizeInit	PositiveInteger	0..1	attr	Size of the monitoring window of state Init for the E2E state machine.
windowSize Invalid	PositiveInteger	0..1	attr	Size of the monitoring window of state Invalid for the E2E state machine.
windowSize Valid	PositiveInteger	0..1	attr	Size of the monitoring window of state Valid for the E2E state machine.

Table 10.59: E2EProfileConfiguration

Enumeration	**DataIdModeEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer
Note	Supported inclusion modes to include the implicit two-byte Data ID in the one-byte CRC.
Literal	*Description*
all16Bit	Two bytes are included in the CRC (double ID configuration). **Tags:**atp.EnumerationLiteralIndex=0
lower12Bit	The low byte is included in the implicit CRC calculation, the low nibble of the high byte is transmitted along with the data (i.e. it is explicitly included), the high nibble of the high byte is not used. This is applicable for the IDs up to 12 bits. **Tags:**atp.EnumerationLiteralIndex=2

Table 10.60: DataIdModeEnum

Class	**E2EProfileCompatibilityProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	This meta-class collects settings for configuration of the E2E state machine. **Tags:**atp.recommendedPackage=E2EProfileCompatibilityPropsCollection			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
transitToInvalid Extended	Boolean	0..1	attr	E2E State machine behavior concerning transition from NODATA/INIT to INVALID value=0 (false): no direct transition from NODATA to INVALID, no transition from INIT to INVALID due to counter-related faults (Autosar R19-11 or former behavior) value=1 (true): direct transition from NODATA to INVALID covered, transition from INIT to INVALID due to counter-related faults covered (state machine extended)

Table 10.61: E2EProfileCompatibilityProps

Please note that the configuration of the E2E state machines with the configuration attributes available in `E2EProfileConfiguration` is restricted by [constr_3176], [constr_3177], [constr_3178], [constr_3179], [constr_3180], [constr_3181] defined in the E2E Protocol specification [29].

It is possible to overwrite the E2E state machine configuration settings that are defined in `E2EProfileConfiguration` at the `RPortPrototype` of a `SwComponentType` with settings available in the `ReceiverComSpec` as described in [TPS_MANI_03132].

With this approach it is possible to define individual E2E settings for different receivers of the event, or field notifiers.

Likewise, it is possible to overwrite the E2E state machine configuration settings that are defined in `E2EProfileConfiguration` at the `RPortPrototype` of a `SwComponentType` with settings available in the `ClientComSpec` as described in [TPS_MANI_01324].

With this approach it is possible to define individual E2E settings for different callers of a `method`.

Finally, it is possible to overwrite the E2E state machine configuration settings that are defined in `E2EProfileConfiguration` at the `PPortPrototype` of a `SwComponentType` with settings available in the `ServerComSpec` as described in [TPS_MANI_01325].

[constr_3493]{DRAFT} **Applicable attributes for standardized E2E Profiles** ⌈ Table 10.62 defines the applicable attributes for the standardized E2E Profiles of AUTOSAR.⌋ *()*

E2E Attributes	Root Element			Attribute Existence per Profile									
	End2EndEventProtectionProps	End2EndMethodProtectionProps	E2EProfileConfiguration	PROFILE_04	PROFILE_05	PROFILE_06	PROFILE_07	PROFILE_08	PROFILE_11	PROFILE_22	PROFILE_04m	PROFILE_07m	PROFILE_44
dataId	x	x		1	1	1	1	1	1	n	1	1	1
dataLength	x	x			x				x	x			
minDataLength	x	x		x		x	x	x			x	x	x
maxDataLength	x	x		x		x	x	x			x	x	x
dataUpdatePeriod	x	x		x	x	x	x	x	x	x	x	x	x
sourceId		x									x	x	
dataIdMode			x						x				
maxDeltaCounter			x	x	x	x	x	x	x	x	x	x	x
maxErrorStateInit			x	x	x	x	x	x	x	x	x	x	x
maxErrorStateInvalid			x	x	x	x	x	x	x	x	x	x	x
maxErrorStateValid			x	x	x	x	x	x	x	x	x	x	x
minOkStateInit			x	x	x	x	x	x	x	x	x	x	x
minOkStateInvalid			x	x	x	x	x	x	x	x	x	x	x
minOkStateValid			x	x	x	x	x	x	x	x	x	x	x
windowSizeValid			x	x	x	x	x	x	x	x	x	x	x
windowSizeInvalid			x	x	x	x	x	x	x	x	x	x	x

▽

△

E2E Attributes	Root Element			Attribute Existence per Profile									
	End2EndEventProtectionProps	End2EndMethodProtectionProps	E2EProfileConfiguration	PROFILE_04	PROFILE_05	PROFILE_06	PROFILE_07	PROFILE_08	PROFILE_11	PROFILE_22	PROFILE_04m	PROFILE_07m	PROFILE_44
windowSizeInit			x	x	x	x	x	x	x	x	x	x	x
clearFromValidToInvalid			x	x	x	x	x	x	x	x	x	x	x

Table 10.62: Allowed Attributes for standardized E2E Profiles

In PROFILE_22 the dataId is defined as a list of 16 dataId values, where a different value is transmitted depending on the counter value.

Please also note that the Classic Platform attributes counterOffset, crcOffset and dataIdNibbleOffset are not configurable in Adaptive Autosar and are set to fixed values by the AUTOSAR Standard.

[constr_5230]{DRAFT} **Existence of attribute E2EProfileCompatibilityProps.transitToInvalidExtended is mandatory for each E2EProfileConfiguration** ⌈For each E2EProfileConfiguration, a reference to E2EProfileCompatibilityProps in the role e2eProfileCompatibilityProps shall exist and the referenced E2EProfileCompatibilityProps shall define a value for the attribute transitToInvalidExtended.⌋*()*

10.4 Secure Communication

AUTOSAR supports different protocols that provide communication security over a network. To configure the secured communication of ServiceInterface elements between a ProvidedApServiceInstance and a RequiredApServiceInstance the ServiceInterfaceElementSecureComConfig meta-class is defined.

[TPS_MANI_03133]{DRAFT} **Usage of ServiceInterfaceElementSecureComConfig** ⌈The ServiceInterfaceElementSecureComConfig element is used to define ServiceInterface element specific secure communication configuration settings in the context of an AdaptivePlatformServiceInstance.⌋*(RS_MANI_00036)*

The modeling allows protecting selected elements of a ServiceInterface, like particular events or methods.

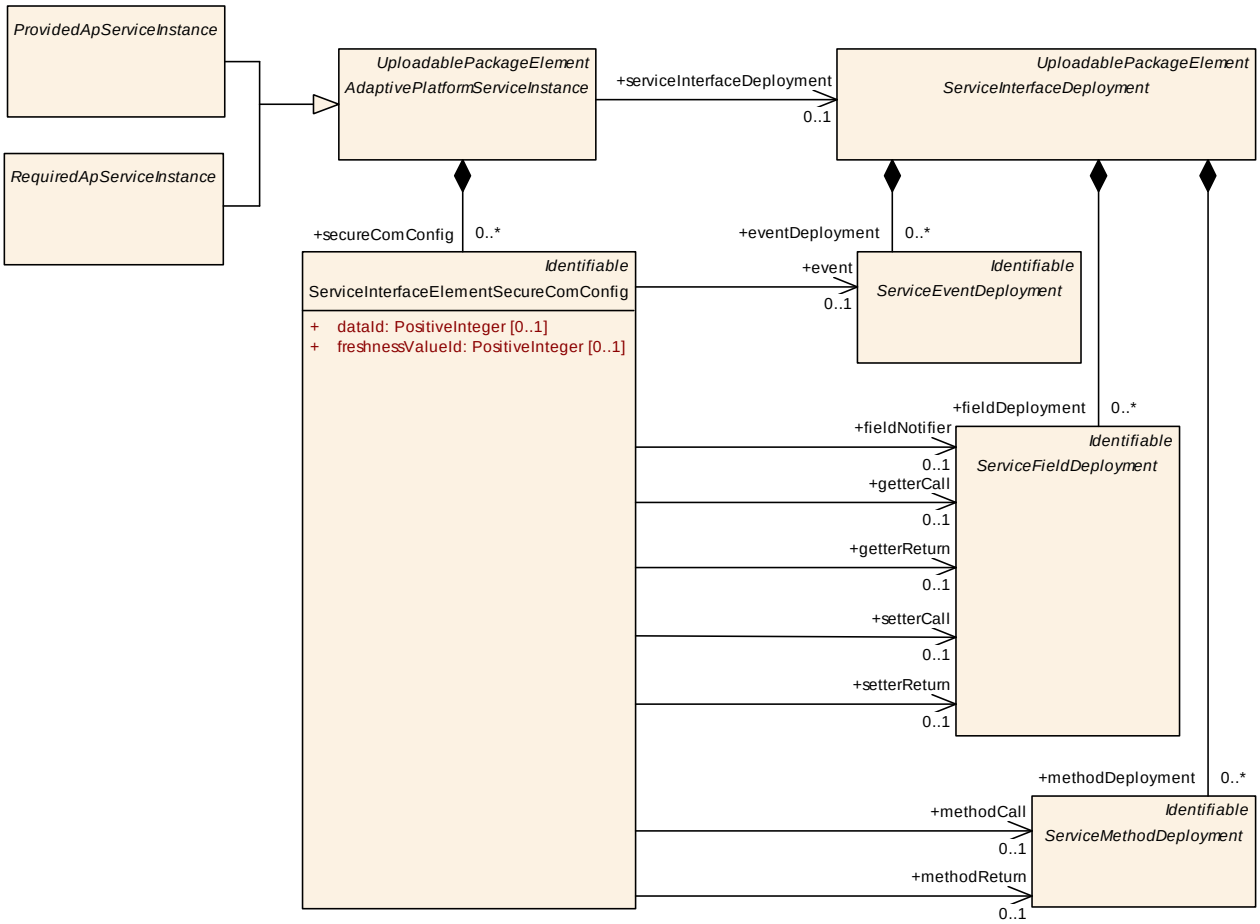


Figure 10.33: Secure Communication

Since the `ServiceInterfaceElementSecureComConfig` meta-class is aggregated by the abstract `AdaptivePlatformServiceInstance` it can be used to configure the secure communication on specific derived classes like `ProvidedSomeipServiceInstance` or `RequiredSomeipServiceInstance` that fit the underlying middleware. With this approach it is possible to define different communication security protections for different used transport layer mechanisms in case of Multi-Binding.

Class	**ServiceInterfaceElementSecureComConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	This element allows to secure the communication of the referenced ServiceInterface element. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataId	PositiveInteger	0..1	attr	This attribute defines a unique numerical identifier for the referenced ServiceInterface element.

△

Class	ServiceInterfaceElementSecureComConfig			
event	ServiceEvent Deployment	0..1	ref	Reference to an event that is protected by a security protocol. **Tags:**atp.Status=draft
fieldNotifier	ServiceField Deployment	0..1	ref	Reference to a field notifier that is protected by a security protocol. **Tags:**atp.Status=draft
freshnessValue Id	PositiveInteger	0..1	attr	This attribute defines the Id of the Freshness Value.
getterCall	ServiceField Deployment	0..1	ref	Reference to a field getter call message that is protected by a security protocol. **Tags:**atp.Status=draft
getterReturn	ServiceField Deployment	0..1	ref	Reference to a field getter return message that is protected by a security protocol. **Tags:**atp.Status=draft
methodCall	ServiceMethod Deployment	0..1	ref	Reference to a method call message that is protected by a security protocol. **Tags:**atp.Status=draft
methodReturn	ServiceMethod Deployment	0..1	ref	Reference to a method return message that is protected by a security protocol. **Tags:**atp.Status=draft
setterCall	ServiceField Deployment	0..1	ref	Reference to a field setter call message that is protected by a security protocol. **Tags:**atp.Status=draft
setterReturn	ServiceField Deployment	0..1	ref	Reference to a field setter return message that is protected by a security protocol. **Tags:**atp.Status=draft

Table 10.63: ServiceInterfaceElementSecureComConfig

[constr_3391]{DRAFT} **`ServiceInterfaceElementSecureComConfig` references to `ServiceInterfaceDeployment` elements** ⌈`ServiceInterfaceElementSecureComConfig` element shall be defined for exactly one `ServiceInterface` element and shall therefore contain only one single reference to an element defined in the scope of a `ServiceInterfaceDeployment`.⌋*()*

The attributes in the `ServiceInterfaceElementSecureComConfig` meta-class are defining security configuration settings that are specific for the referenced `ServiceInterface` element in the context of an `AdaptivePlatformServiceInstance`. The used security protocol is defined in the `ServiceInstanceToMachineMapping`.

[TPS_MANI_03199]{DRAFT} **Endpoint protection by `SecureComProps`** ⌈The `ServiceInstanceToMachineMapping` allows to assign a security protocol configuration settings that are defined in the referenced `SecureComProps` meta-class to protect endpoints that are defined by the Transport Protocol, Port and IP Address on which one or several `AdaptivePlatformServiceInstance`s are provided or consumed.⌋*(RS_MANI_00036)*

[TPS_MANI_03200]{DRAFT} **`SecureComProps` for udp, tcp and multicast communication** ⌈The `ServiceInstanceToMachineMapping` allows to assign a security protocol configuration settings for:

- udp communication if the ServiceInstanceToMachineMapping refers to the SecureComProps in the role secureComPropsForUdp
- tcp communication in case the ServiceInstanceToMachineMapping refers to the SecureComProps in the role secureComPropsForTcp
- multicast communication in case the ServiceInstanceToMachineMapping refers to the SecOcSecureComProps in the role secOcComPropsForMulticast

⌋(*RS_MANI_00036*)

Please note that protection of IP multicast traffic is only supported by SecOC and therefore the ServiceInstanceToMachineMapping refers directly the SecOcSecureComProps in the secOcComPropsForMulticast role.

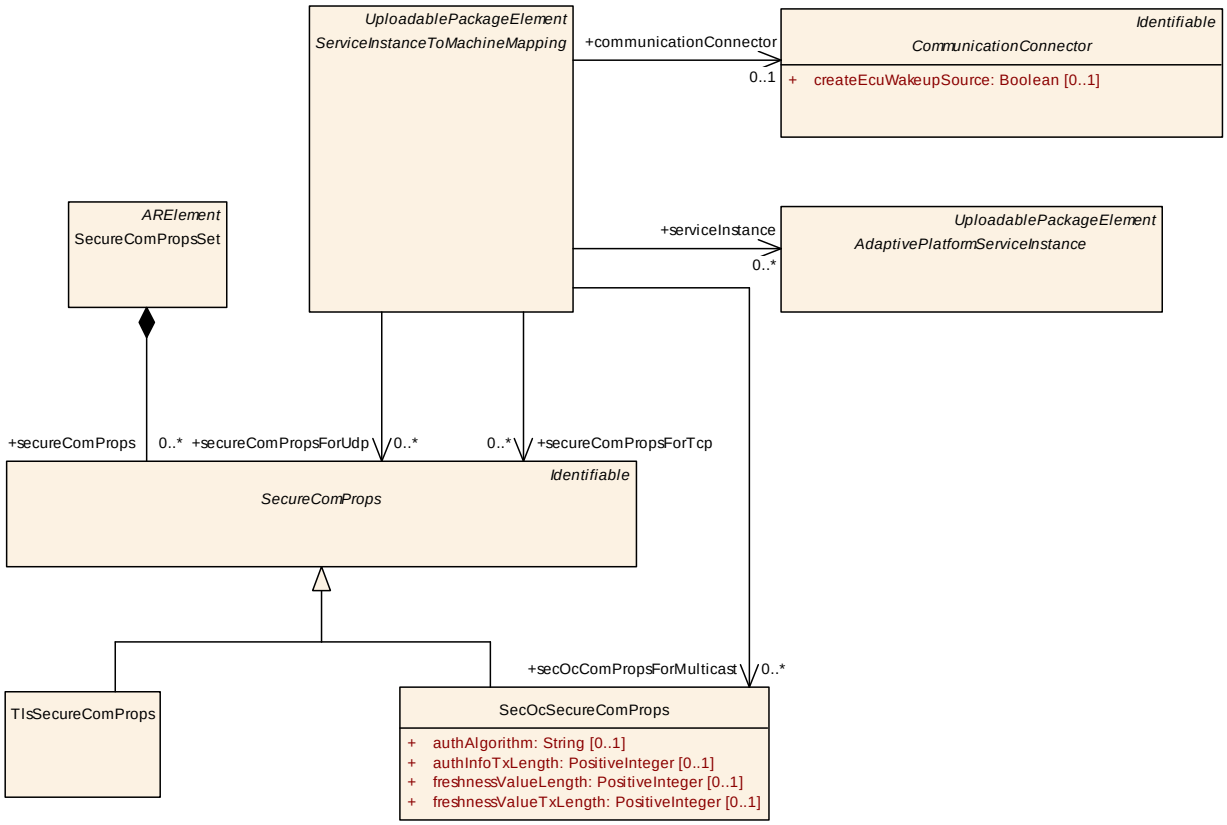


Figure 10.34: Security protocol configuration

Class	**SecureComPropsSet**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication
Note	This meta-class represents the ability to aggregate a collection of SecureComProps.. **Tags:** atp.Status=draft atp.recommendedPackage=SecureComPropsSets

▽

△

Class	SecureComPropsSet			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, Packageable Element, Referrable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
secureCom Props	SecureComProps	*	aggr	This represents the collection of SecureComProps aggregated at the SecureComPropsSet. **Tags:**atp.Status=draft

Table 10.64: SecureComPropsSet

Class	***SecureComProps*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	This meta-class defines a communication security protocol and its configuration settings. **Tags:**atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Subclasses	SecOcSecureComProps, TlsSecureComProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 10.65: SecureComProps

10.4.1 Secure Communication over TLS

The configuration of the Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) protocols is supported with the `TlsSecureComProps` meta-class, which is a specialization of `SecureComProps`.

It is a common use case that only one end of a TLS-based connection is actually modeled in an AUTOSAR model. It is therefore important that the modeling does not rely on or imply knowledge about both ends of such a TLS-based connection.

An AUTOSAR model that only describes one end of the communication is positively required to work, independently of the availability of a formal modeling of the other end.

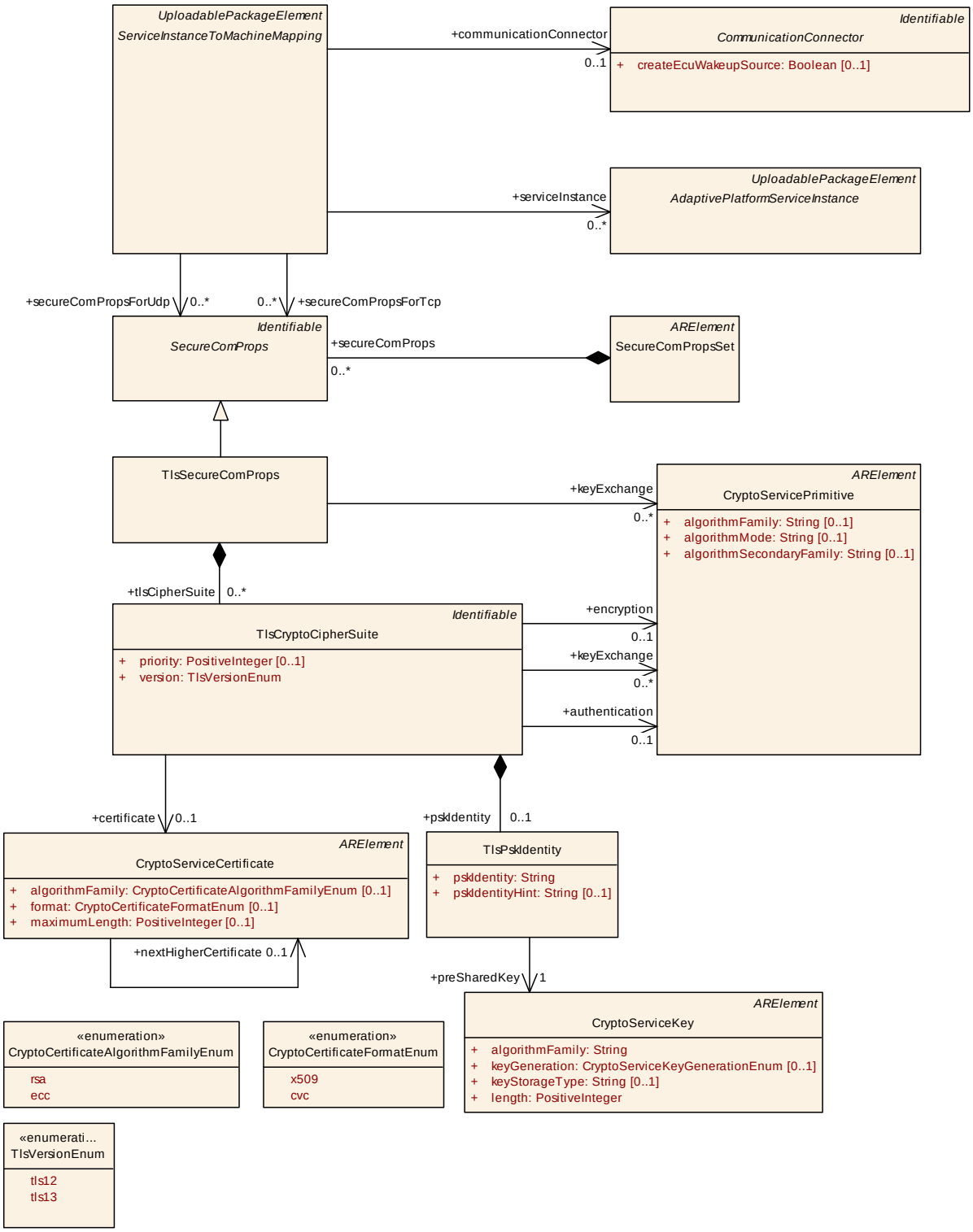


Figure 10.35: Secure Communication over TLS

Class	**TlsSecureComProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	Configuration of the Transport Layer Security protocol (TLS). **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SecureComProps*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
keyExchange	CryptoServicePrimitive	*	ref	This reference identifies the shared (i.e. applicable for each of the aggreated cipher suites) crypto service primitive for the execution of key exchange during the handshake phase. **Tags:**atp.Status=draft
tlsCipherSuite	TlsCryptoCipherSuite	*	aggr	Collection of supported cipher suites that are used to negotiate the security settings for a network connection defined by the ServiceInstanceToMachineMapping. **Tags:**atp.Status=draft

Table 10.66: TlsSecureComProps

Enumeration	**CryptoServiceKeyGenerationEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This enumeration shall be taken to express the handling of a crypto key in terms of whether it is obtained from e.g. a diagnostic tester or whether it is created by derivation from a master key.
Literal	***Description***
keyDerivation	This means that the crypto key is created by derivation from a master key. **Tags:**atp.EnumerationLiteralIndex=0
keyStorage	This means that the crypto key is obtained from an external entity, e.g. a diagnostic tester. **Tags:**atp.EnumerationLiteralIndex=1

Table 10.67: CryptoServiceKeyGenerationEnum

TLS is composed of the TLS Record Protocol and the TLS Handshake Protocol. The Record Protocol provides connection security and encrypts and authenticate packets. The record layer functions can be called at any time after the handshake process is finished, when there is need to receive or send data.

The Handshake Protocol allows the server and client to authenticate each other and to negotiate encryption algorithms and cryptographic keys before any data is exchanged.

In order to establish a cryptographically secure data channel, the communication partners in form of `ServiceInstanceToMachineMapping`s shall agree on ciphersuites and on keys that will be used to encrypt the data.

The client sends a list of supported ciphersuites to the server. The server decides on a ciphersuite from the list provided by the client, and continues with the handshake. Please note that the server and client roles cannot be swapped while the connection exists, i.e. a *server* remains the *server* for the full amount of time the connection exists.

[TPS_MANI_03213]{DRAFT} **Semantics of meta-class `TlsSecureComProps`** ⌈As a sub-class of `SecureComProps`, meta-class `TlsSecureComProps` has the ability to

collect the TLS-related configuration aspects from either the perspective of the client or the server.

In the case of TLS, the collection boils down to the aggregation of meta-class `TlsCryptoCipherSuite` in the role `tlsCipherSuite` plus the ability (by means of the role `keyExchange`) to define handshake properties that are shared for each of the aggregated `tlsCipherSuite`.⌋*(RS_MANI_00036)*

[constr_5047]{DRAFT} **Supported values of `ServiceInstanceToMachineMapping.category`** ⌈The only supported values of attribute `TlsSecureComProps.category` are:

- **`TLS_SERVER`**: the `TlsSecureComProps` assumes the role of the *server* in the TLS connection.
- **`TLS_CLIENT`**: the `TlsSecureComProps` assumes the role of the *client* in the TLS connection.

⌋*()*

[TPS_MANI_03134]{DRAFT} **Configuration of supported TLS ciphersuites** ⌈The creation of a TLS connection requires the usage of a suite of cryptographic operations in specific roles, also known as a *cipher suite*.

Meta-class `TlsCryptoCipherSuite` represents a given cipher suite for a TLS connection. `TlsCryptoCipherSuite` references meta-class `CryptoServicePrimitive` in three dedicated roles that represent the steps of the creation of a TLS connection.

More specifically, the cryptographic operations for setting up a TLS connection involve the following steps:

- **Key exchange**: these `CryptoServicePrimitive`s may be used for the handshake phase of the TLS connection. Different alternatives exist for executing this phase and therefore the multiplicity of this reference is 0..*.
- **Authentication** of communication partners during the operational phase of the TLS connection. For this purpose a single `CryptoServicePrimitive` is used on each end of the communication.
- **Encryption** of content exchanged between the communication partners that have established the TLS connection. For this purpose a single `CryptoServicePrimitive` is used on each end of the communication.

⌋*(RS_MANI_00036)*

Class	TlsCryptoCipherSuite			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class represents a cipher suite for describing cryptographic operations in the context of establishing a connection of ApplicationEndpoints that is protected by TLS.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
authentication	CryptoServicePrimitive	0..1	ref	This reference identifies the crypto service primitive for the generation and verification of MACs.
certificate	CryptoService Certificate	0..1	ref	This reference identifies the applicable certificate.
encryption	CryptoServicePrimitive	0..1	ref	This reference identifies the crypto service primitive for the execution of encryption.
keyExchange	CryptoServicePrimitive	*	ref	This reference identifies the individual (i.e. per cipher suite) crypto service primitive for the execution of key exchange during the handshake phase.
priority	PositiveInteger	0..1	attr	This attribute identifies the priority of the cipher suite. Range: 1..65535. Lower values represent higher priorities.
pskIdentity	TlsPskIdentity	0..1	aggr	Pre-shared key identity shared during the handshake among the communication parties, to establish a TLS connection if the handshake is based on the existence of a pre-shared key.
version	TlsVersionEnum	1	attr	This attribute supports the definition of the applicable version of TLS.

Table 10.68: TlsCryptoCipherSuite

Class	CryptoServicePrimitive			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class has the ability to represent a crypto primitive. **Tags:**atp.recommendedPackage=CryptoPrimitives			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
algorithmFamily	String	0..1	attr	This attribute represents a description of the family (e.g. AES) of crypto algorithm implemented by the crypto primitive.
algorithmMode	String	0..1	attr	This attribute represents a description of the mode of the crypto algorithm implemented by the crypto primitive.
algorithm Secondary Family	String	0..1	attr	This attribute represents a further description of the secondary family of crypto algorithm implemented by the crypto primitive. The secondary family is needed for the specification of the hash algorithm for a signature check, e.g. using RSA.

Table 10.69: CryptoServicePrimitive

Class	**CryptoServiceKey**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class has the ability to represent a crypto key **Tags:**atp.recommendedPackage=CryptoDevelopmentKeys			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
algorithmFamily	String	1	attr	This attribute represent the description of the family of the applicable crypto algorithm.
development Value	ValueSpecification	0..1	aggr	This aggregation represents the ability to assign a specific value to the crypto key as part of the system description. This value can then be taken for the development of the respective ECU.
keyGeneration	CryptoServiceKey GenerationEnum	0..1	attr	This attribute describes how a the specific cryptographic key is created.
keyStorageType	String	0..1	attr	This attribute describes where the enclosing cryptographic key shall be stored. AUTOSAR reserves specific values for this attributes but it is possible to insert custom values as well.
length	PositiveInteger	1	attr	This attribute describes the length of the cryptographic key.

Table 10.70: CryptoServiceKey

[TPS_MANI_03214]{DRAFT} **Existence of `TlsCryptoCipherSuite.keyExchange` vs. `TlsSecureComProps.keyExchange`** ⌈The role `TlsSecureComProps.keyExchange` has been introduced as an optimization.

It is assumed that the references for key exchange look pretty similar if not identical for many concrete `TlsCryptoCipherSuite`s.

Adding these references in an identical form to a bunch of `TlsCryptoCipherSuite`s does not really make sense. Therefore, `TlsSecureComProps` allows to define these references as well with the intention to make them valid for all `TlsSecureComProps.tlsCipherSuite`s.

A mixture of references in the role `TlsCryptoCipherSuite.keyExchange` and `TlsSecureComProps.keyExchange` is supported.⌋*(RS_MANI_00036)*

[TPS_MANI_03215]{DRAFT} **Semantics of `CryptoServiceCertificate`** ⌈Meta-class `CryptoServiceCertificate` represents a cryptographic certificate needed for the creation of a TLS connection between *server* and *client*.⌋*()*

Class	**CryptoServiceCertificate**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This meta-class represents the ability to model a cryptographic certificate. **Tags:**atp.recommendedPackage=CryptoServiceCertificates
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*

▽

△

Class	**CryptoServiceCertificate**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
algorithmFamily	CryptoCertificate AlgorithmFamilyEnum	0..1	attr	This attribute represents a description of the family of crypto algorithm used to generate public key and signature of the cryptographic certificate.
format	CryptoCertificateFormat Enum	0..1	attr	This attribute can be used to provide information about the format used to create the certificate
maximum Length	PositiveInteger	0..1	attr	This attribute represents the ability to define the maximum length of the certificate.
nextHigher Certificate	CryptoService Certificate	0..1	ref	The reference identifies the next higher certificate in the certificate chain.

Table 10.71: CryptoServiceCertificate

Enumeration	**CryptoCertificateAlgorithmFamilyEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This meta-class defies possible cryptographic algorithm families used to create public keys and signatures within the certificate.
Literal	***Description***
ecc	The cryptographic operations in the certificate are executed using elliptic curves (ecc) **Tags:**atp.EnumerationLiteralIndex=2
rsa	The cryptographic operations in the certificate are executed using the RSA approach. **Tags:**atp.EnumerationLiteralIndex=1

Table 10.72: CryptoCertificateAlgorithmFamilyEnum

Enumeration	**CryptoCertificateFormatEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This meta-class defines possible formats of cryptographic certificates.
Literal	***Description***
cvc	The certificate has been created in Card Verifiable Certificate (CVC) format **Tags:**atp.EnumerationLiteralIndex=2
x509	The certificate is created in X.509 format. **Tags:**atp.EnumerationLiteralIndex=1

Table 10.73: CryptoCertificateFormatEnum

[constr_5048]{DRAFT} **Existence of `TlsCryptoCipherSuite.certificate` and `TlsCryptoCipherSuite.pskIdentity` in the *server* role** ⌈**Either**

- the reference to `CryptoServiceCertificate` in the role `TlsCryptoCipherSuite.certificate`
- the aggregation of `TlsPskIdentity` in the role `TlsCryptoCipherSuite.pskIdentity`

shall exist if the `TlsCryptoCipherSuite` is aggregated by `TlsSecureComProps` that has the attribute `category` set to the value `TLS_SERVER`.⌋*()*

In other words two different approaches are supported by TLS for the handling of key compromise: Pre-shared secret and certificate.

The server may optionally request a certificate from the *client*. If this option is not used then other documented approaches for completing the handshake phase is foreseen for the specific case.

[TPS_MANI_03216]{DRAFT} **Existence of `TlsCryptoCipherSuite.certificate` and `TlsCryptoCipherSuite.pskIdentity` in the *client* role** ⌈The client (`TlsSecureComProps` has set the value of attribute `category` to `TLS_CLIENT`) has the following authentication options:

- the reference to `CryptoServiceCertificate` in the role `TlsCryptoCipherSuite.certificate` exists,
- the aggregation of `TlsPskIdentity` in the role `TlsCryptoCipherSuite.pskIdentity` exists,
- neither one nor the other exists. In this case the handshake is provided on the basis of the server certificate only.

⌋*(RS_MANI_00036)*

In the pre-shared Key approach the client indicates which key to use by including a `pskIdentity` in the ClientKeyExchange message. To help the client in selecting which identity to use, the server can provide a `pskIdentityHint` in the ServerKeyExchange message. Please note that the usage of `pskIdentityHint`s is restricted for usage with TLS 1.2.

[TPS_MANI_03137]{DRAFT} **`ServiceInterfaceElementSecureComConfig` is not relevant in case of TLS communication** ⌈The element `ServiceInterfaceElementSecureComConfig` is not relevant in case of TLS communication.⌋ *(RS_MANI_00036)*

[constr_3485]{DRAFT} **UDP endpoint using DTLS can only serve provided or required service instances exclusively** ⌈A `ServiceInstanceToMachineMapping` that refers to `TlsSecureComProps` in the role `secureComPropsForUdp` is not allowed to refer to `ProvidedApServiceInstance`s and at the same time to `RequiredApServiceInstance`s in the role `serviceInstance`. In other words a UDP endpoint using DTLS can only serve provided or required service instances exclusively.⌋ *()*

[constr_3486]{DRAFT} **TCP endpoint using TLS can only serve provided or required service instances exclusively.** ⌈A `ServiceInstanceToMachineMapping` that refers to `TlsSecureComProps` in the role `secureComPropsForTcp` is not allowed to refer to `ProvidedApServiceInstance`s and at the same time to `RequiredApServiceInstance`s in the role `serviceInstance`. In other words a TCP endpoint using TLS can only serve provided or required service instances exclusively.⌋ *()*

The reason for [constr_3485] and [constr_3486] is that the (D)TLS client needs to establish the (D)TLS connection and a TCP/UDP endpoint that is described by the `ServiceInstanceToMachineMapping` can only take one role: (D)TLS client or (D)TLS server. If a `ServiceInstanceToMachineMapping` would act as (D)TLS client and

would refer to a ProvidedApServiceInstance then this (D)TLS client would need to establish the (D)TLS connection. But in this case the (D)TLS client would not know to which remote service client a connection needs to be established since different RequiredApServiceInstances may directly call methods of the ProvidedApServiceInstance without any registration.

The same issue exists if the ServiceInstanceToMachineMapping acts as (D)TLS server and refers to RequiredApServiceInstances. The (D)TLS client needs to establish the (D)TLS connection before any messages are exchanged. But the remote service provider has no knowledge that this service consumer wants to call methods over a (D)TLS connection.

10.4.2 Secure Communication over SecOC

AUTOSAR Secure Onboard Communication (SecOC) supports symmetric and asymmetric authentication approaches. To configure the SecOC secure protection of a message by a MAC or Signature the ServiceInterfaceElementSecureComConfig needs to be defined. This element contains the configuration settings for the individual ServiceInterface elements. In addition, the ServiceInstanceToMachineMapping needs to point to SecOcSecureComProps to configure the endpoint protection that is defined by the Transport Protocol, Port and IP Address.

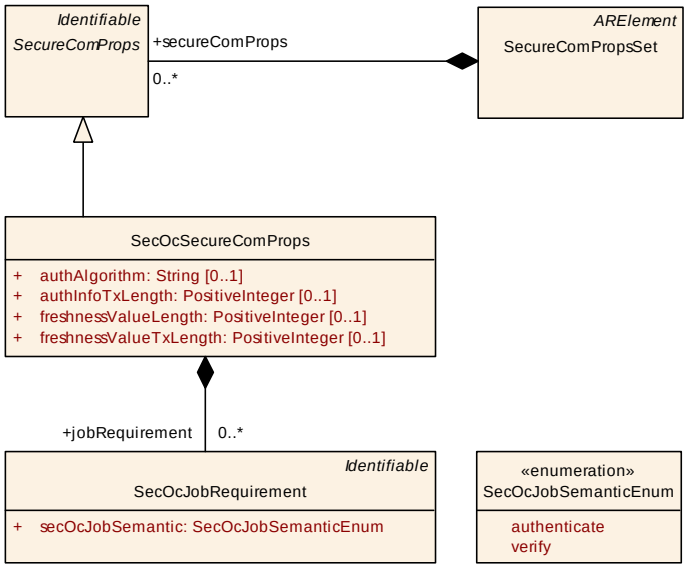


Figure 10.36: Secure Communication over SecOC

[constr_3392]{DRAFT} **ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are mandatory in case of SecOC communication** ⌈The attributes ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are mandatory in case of SecOC communication.⌋ *()*

[TPS_MANI_03138]{DRAFT} **SecOC Security Profile** ⌈The SecOC security profile is defined by `SecOcSecureComProps.category`.⌋*(RS_MANI_00036)*

Class	**SecOcSecureComProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	Configuration of AUTOSAR SecOC. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SecureComProps*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authAlgorithm	String	0..1	attr	This attribute defines the authentication algorithm used for MAC generation and verification.
authInfoTx Length	PositiveInteger	0..1	attr	This attribute defines the length in bits of the authentication code to be included in the payload of the authenticated Message.
freshnessValue Length	PositiveInteger	0..1	attr	This attribute defines the complete length in bits of the Freshness Value.
freshnessValue TxLength	PositiveInteger	0..1	attr	This attribute defines the length in bits of the Freshness Value to be included in the payload of the secured message. In other words this attribute defines the length of the authenticated Message.
jobRequirement	SecOcJobRequirement	*	aggr	Collection of cryptographic job requirements. **Tags:**atp.Status=draft

Table 10.74: SecOcSecureComProps

[TPS_MANI_03139]{DRAFT} **Standardized SecOC Security Profiles** ⌈The SecOC security profile that is defined by `SecOcSecureComProps.category` can have the following values that are standardized by AUTOSAR: PROFILE_01, PROFILE_02, PROFILE_03.⌋*(RS_MANI_00036)*

The attribute values for the predefined categories mentioned in [TPS_MANI_03139] are defined in [constr_3325] in [17].

[TPS_MANI_03140]{DRAFT} **Non-Standardized SecOC Security Profiles** ⌈The values for the `SecOcSecureComProps.category` mentioned in [TPS_MANI_03139] are standardized and reserved for being used in the way the AUTOSAR standard foresees. In addition, it is positively possible to use other than the standardized values for the `SecOcSecureComProps.category`.⌋*(RS_MANI_00036)*

With the `SecOcJobRequirement` the cryptographic routines can be selected that need to be supported. In case of SecOC it can be selected whether the symmetric and/or asymmetric authentication approach is needed.

Class	**SecOcJobRequirement**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication
Note	Requirements for the cryptographic job that need to be executed. **Tags:**atp.Status=draft

▽

△

Class	**SecOcJobRequirement**			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
secOcJob Semantic	SecOcJobSemantic Enum	1	attr	This attribute defines the cryptographic algorithm that needs to be supported.

Table 10.75: SecOcJobRequirement

Enumeration	**SecOcJobSemanticEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication
Note	List of cryptographic routines supported by SecOC. **Tags:**atp.Status=draft
Literal	***Description***
authenticate	Authentication algorithm for Authenticator generation/verification. **Tags:**atp.EnumerationLiteralIndex=0
verify	Asymmetric cryptographic algorithm to generate/verify a signature **Tags:**atp.EnumerationLiteralIndex=1

Table 10.76: SecOcJobSemanticEnum

10.5 Raw Data Stream

10.5.1 Raw Data Stream Deployment

[TPS_MANI_01285]{DRAFT} **Purpose of meta-class `RawDataStreamDeployment`** ⌈Meta-class `RawDataStreamDeployment` has the ability to further qualify an existing `AbstractRawDataStreamInterface` on deployment level.⌋*(RS_MANI_00067)*

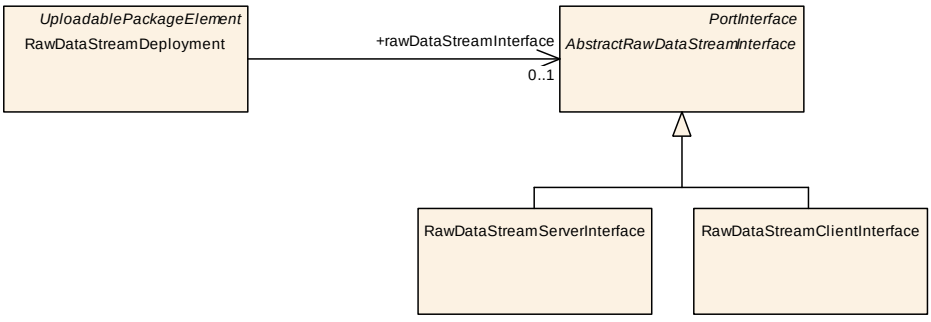


Figure 10.37: Modeling of the `RawDataStreamDeployment`

Class	RawDataStreamDeployment			
Package	M2::AUTOSARTemplates::AdaptivePlatform::RawDataStreamMapping			
Note	This meta-class represents the ability to model deployment-level information for a raw data stream **Tags:** atp.Status=draft atp.recommendedPackage=RawDataStreamDeployments			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
rawDataStream Interface	AbstractRawData StreamInterface	0..1	ref	This reference identifies the corresponding RawData StreamInterface, **Tags:**atp.Status=draft

Table 10.77: RawDataStreamDeployment

10.5.2 Raw Data Stream Mapping

[TPS_MANI_01287]{DRAFT} **Semantics of `RawDataStreamMapping`** ⌈On the deployment side, the access to a raw data stream requires the provision of actual transport for the raw data.

In principle, it would be possible to implement the transport on top of various technologies.

Therefore, abstract meta-class `RawDataStreamMapping` exists to provide the principle ability to map to an `RPortPrototype` and to a `Process` on the one hand.

The mapping to a concrete transport technology is left to sub-classes of `RawDataStreamMapping`, for example `EthernetRawDataStreamMapping`.⌋*(RS_MANI_00067)*

Class	***RawDataStreamMapping*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::RawDataStreamMapping			
Note	This meta-class acts as an abstract base class for mapping raw data streams to the application software. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Subclasses	EthernetRawDataStreamMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
deployment	RawDataStream Deployment	0..1	ref	This reference identifies the applicable RawDataStream Deployment. **Tags:**atp.Status=draft

▽

△

Class	***RawDataStreamMapping*** (abstract)			
portPrototype	RPortPrototype	0..1	iref	Reference to a specific PortPrototype that represents the raw data stream to the application. **Tags:**atp.Status=draft **InstanceRef implemented by:**RPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	Reference to the Process in which the Executable that contains the SoftwareComponent and the referenced Port Prototype is executed. **Tags:**atp.Status=draft

Table 10.78: RawDataStreamMapping

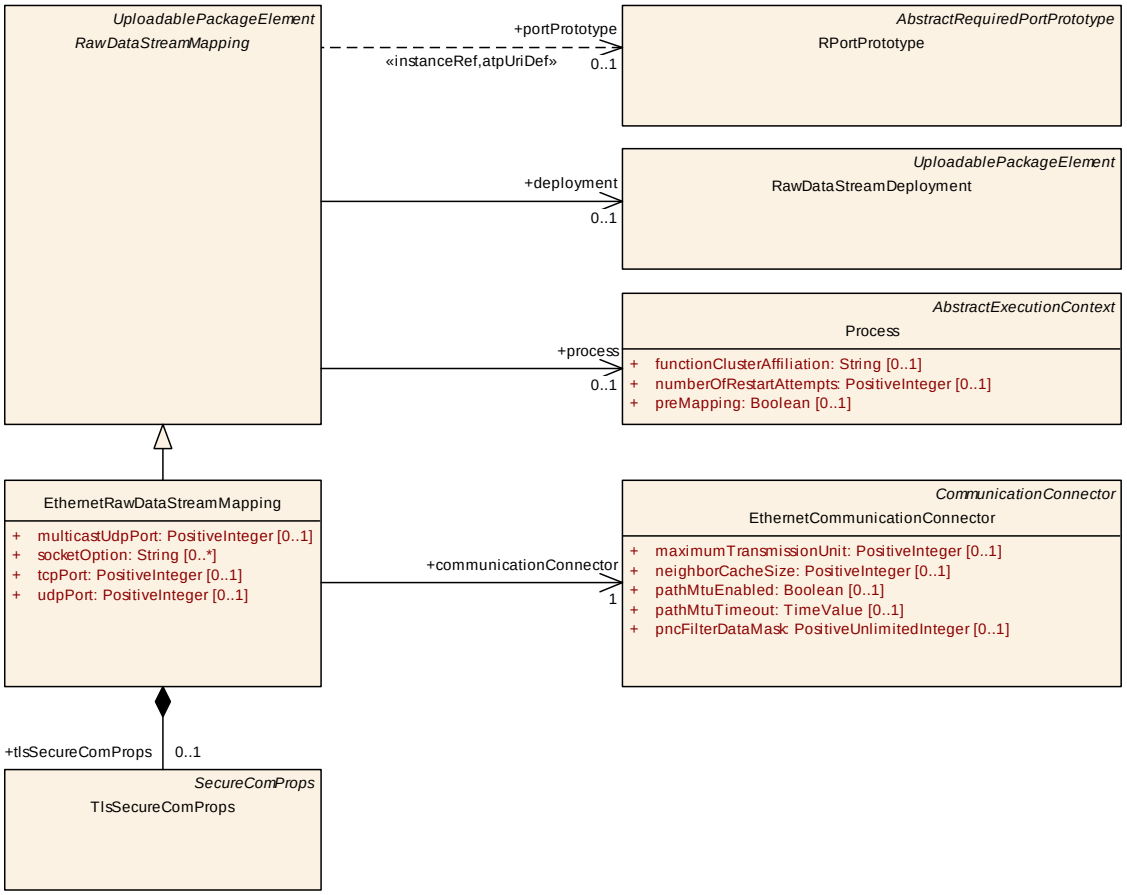


Figure 10.38: Modeling of the `RawDataStreamMapping`

The modeling abstraction of a socket is represented by meta-class `Communication-Connector`, specifically the subclass `EthernetCommunicationConnector`.

Class	**EthernetRawDataStreamMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::RawDataStreamMapping			
Note	This meta-class represents the ability to map a PortPrototype to a Ethernet-based communication channel. **Tags:** atp.Status=draft atp.recommendedPackage=SocketRawDataStreamingMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communicationConnector	EthernetCommunicationConnector	1	ref	This attribute represents the CommunicationConnector taken for socket-based data communication. **Tags:**atp.Status=draft
multicastUdpPort	PositiveInteger	0..1	attr	This attribute describes the UDP Port used for multicast raw data stream transmission.
socketOption	String	*	attr	This attribute represents the ability to specify non-formal socket options that might only be valid for specific platforms. AUTOSAR does not define a standardized meaning for the possible values of this attribute.
tcpPort	PositiveInteger	0..1	attr	This attribute describes the TCP port used for the raw data streaming.
tlsSecureComProps	TlsSecureComProps	0..1	aggr	This aggregation provides the ability to define TLS-related properties for the enclosing SocketRawDataStreamMapping. **Tags:**atp.Status=draft
udpPort	PositiveInteger	0..1	attr	This attribute describes the UDP Port used for the raw data streaming.

Table 10.79: EthernetRawDataStreamMapping

11 Signal-based communication

11.1 Overview

The applications on the adaptive platform communicate with each other in a service-oriented manner. But there is also a use case where applications on the *AUTOSAR adaptive platform* need to communicate with software-components running on the *AUTOSAR classic platform*.

If the remote ECU on the *AUTOSAR classic platform* communicates via SOME/IP in a service-oriented manner and uses the SOME/IP transformer to serialize its data, then the communication with the `Machine` on the *AUTOSAR adaptive platform* can be established directly without any adaptations of neither the ECU nor the `Machine`.

If the counterpart on the *AUTOSAR classic platform* ECU communicates using signal-based communication over, e.g., CAN or FlexRay, the translation of the signal-based content into `ServiceInterface`s needs to be established. The preconditions for this use-case are defined in section 11.2.

Such a signal service translation may happen in a Gateway that is implemented on an ECU on the *AUTOSAR classic platform*. Such a solution is out of scope of this document since it is handled using the *AUTOSAR classic platform* configuration means. This approach is defined in the System Template specification for the Classic platform [17]. It is up to the vehicle architecture design to choose whether the signal service translation shall be implemented on a Classic platform ECU or on an Adaptive platform Machine.

Another alternative for this translation is to happen directly on the `Machine` on the *AUTOSAR adaptive platform* by an Application that is running in the Process, as sketched in Figure 11.1.

This Application communicates with other applications on the *AUTOSAR adaptive platform* in the service-oriented way over ara::com; but it is also able to transmit and receive `ISignal`s as well as communicate signal-based with remote ECUs on the *AUTOSAR classic platform*.

In order to make this possible, software that conforms to the specification of the COM stack on the *AUTOSAR classic platform* needs to be executed on the `Machine` on the *AUTOSAR adaptive platform*.

For the configuration of this software, the System Description based on the System Template on the *AUTOSAR classic platform* is used that contains a communication matrix description with `Pdu`s and `ISignal`s.

This chapter introduces a modeling that creates a bridge between the service-oriented communication based on `ServiceInterface`s of the *AUTOSAR adaptive platform* and the signal-based communication involving the definition of `Pdu`s and `ISignal`s that are used on the *AUTOSAR classic platform*.

The Signal-to-Service mapping, together with the *AUTOSAR classic platform* System Description, allows to configure the communication between a `Machine` on the *AUTOSAR adaptive platform* and an ECU on the *AUTOSAR classic platform*. Please note that in a setup like the one sketched in Figure 11.1, the *AUTOSAR classic platform* System Description would also contain a Pdu or Signal Gateway configuration between the Ethernet and the CAN network to forward the PDUs between the networks.

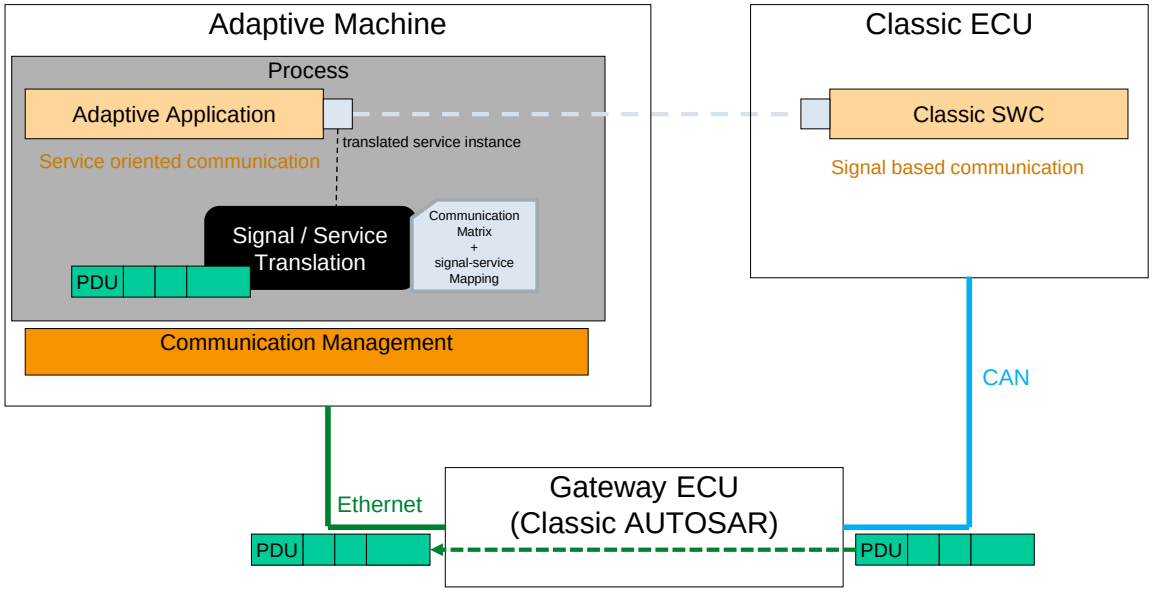


Figure 11.1: SignalToService translation in Application on Adaptive Machine

The *translated service instance* is accessed from the *Adaptive Application* using the ara::com API. The translation is designed as a `network binding` similar to the binding of `SOME/IP`. So the communication direction is

- if the signal-based payload is received by the `Machine` then the Adaptive Application has a `RPortPrototype`
- if the signal-based payload is sent by the `Machine` then the Adaptive Application has a `PPortPrototype`.

Based on the signal-service-translation network binding there are several approaches for further processing the translated information (illustrated in figure 11.2):

- The Adaptive Application directly consumes (produces[1]) the translated service (1)
  This is the typical approach if there is only one Adaptive Application interested in the translated data.
- The Adaptive Application manages and performs a functional routing (2)
  This is the approach if the translated data shall be available as a service again for further processing. Here it is up to the implementation of the Adaptive Application

[1] In the explanations the direction from signal to service is usually used, the service to signal direction is supported as well. For simplicity this is only mentioned explicitly if the mapping behavior is not symmetrical.

how the translated data is routed to the secondary service, especially whether data combination and data conversion are applied.

- Pass Through Connectors: The translated service is passed through to be available to further Adaptive Applications. This is the approach if there exist several further Adaptive Applications which are using the translated service. In this case the translation only has be performed once. Two cases can be distinguished:
  - The translated service and the secondary service use the same `ServiceInterface` (3)
  - The translated service and the secondary service use different (but compatible) `ServiceInterface`s (4)

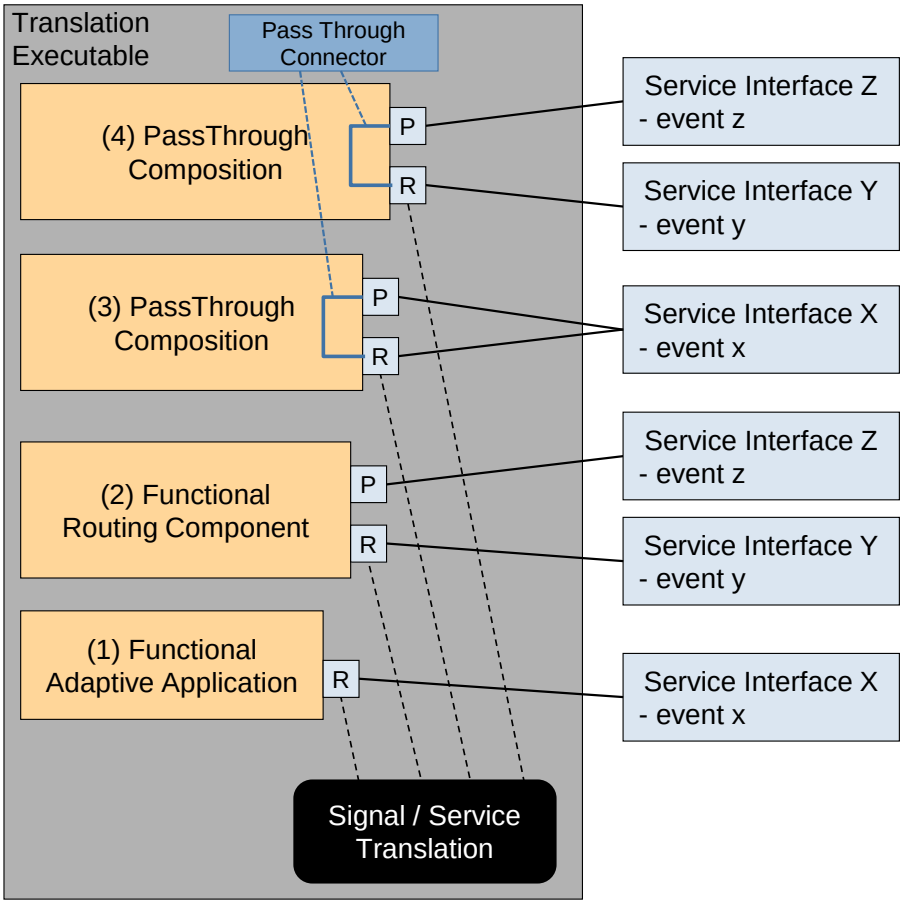


Figure 11.2: Signal-service-translation and PassThroughConnector

The methodological approach is that out of the signal-service mapping descriptions it is possible to automatically generate the network binding code for the signal-service-translation.

If there is a `CompositionSwComponentType` defined with `PassThroughSwConnector`s then also the `AdaptiveApplicationSwComponentType` implementing the pass through behavior can be generated (see section 11.7).

Please note that the configuration of such signal-based communication on an adaptive machine may be solved in two different ways:

1. The communication matrix definition (ARXML System Description) and the Signal-to-Service mapping is available on the target machine and is interpreted at run-time (like the manifest approach).
2. The communication matrix definition (ARXML System Description) and the Signal-to-Service mapping is built off-board and the application executable gets uploaded to the target Machine in response to changes in the communication matrix.

11.2 Signal-based prerequisites

For the signal-service-translation to be available on the adaptive platform the Pdus containing the ISignals have to be accessible on the network which is connected to the adaptive Machine.

As the sole communication network currently supported is Ethernet, the Pdus have to be transported on that network.

Although there is in theory the possibility to directly put the Pdu on the Ethernet this approach would require an individual Socket per Pdu. This is an approach where especially Classic platform ECUs do not have enough resources available to allow individual Sockets per Pdu.

It is required by the signal-service-translation that the Pdus are routed to the Ethernet network using the Header Id feature of SOME/IP (see figure 11.3).

[TPS_MANI_03577]{DRAFT} **headerId required for signal-service-translation** ⌈Every Pdu which shall be processed by signal-service-translation shall have the SocketConnectionIpduIdentifier.headerId defined according to the SOME/-IP definition.⌋*(RS_MANI_00063)*

In order to have efficient transport of many Pdus on Ethernet additionally the SOME/-IP collection mechanism may be used (i.e. definition of pduCollectionTrigger, pduCollectionSemantics, pduCollectionPduTimeout).

In case of service to signal translation the Adaptive Application provides a translated service (defined by a ProvidedSomeipServiceInstance). In this case the content of the associated PDUs shall come fully from that Adaptive Application. That means that the produced PDUs have one source. This is automatically given because the ProvidedSomeipServiceInstance is mapped to one PPortPrototype of one Adaptive Application.

As the content of a PDU is freely composable for signal-based payload there is a possibility that for a sent PDU the contained signals are mapped from different services. In this case the payload of that PDU has to be provided by one Adaptive Application.

Thus, the signal-service-translation is able to compose the PDU payload in the context of that Adaptive Application.

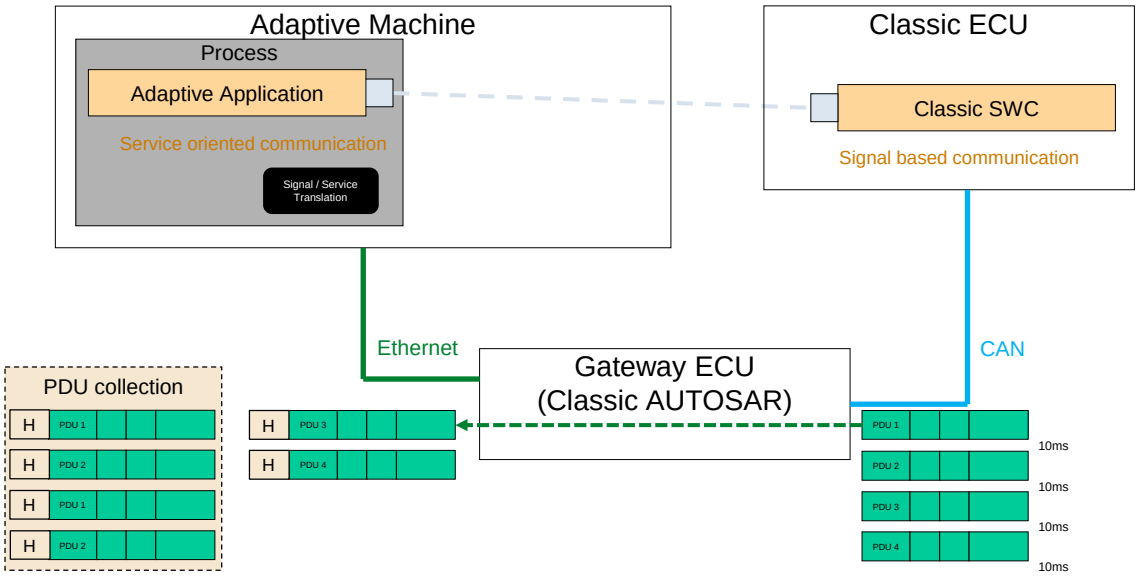


Figure 11.3: Network architecture for signal-service-translation

11.3 Signal-based Deployment

The signal-service-translation is embedded in the SOME/IP deployment using SomeipServiceInterfaceDeployment. The attribute SomeipEventDeployment.serializer defines whether the someip or the signalBased serialization shall be used for a specific event (see also figure 10.3).

[TPS_MANI_03578]{DRAFT} **Signal-based ServiceInterface binding over Ethernet** ⌈In case of signal-based communication over Ethernet the SomeipServiceInterfaceDeployment is used to define the Pdu transport over the network.⌋ *(RS_MANI_00063)*

This aspect is described in section 11.2.

[TPS_MANI_03579]{DRAFT} **Signal-based ServiceEventDeployment over Ethernet** ⌈If the attribute SomeipEventDeployment.serializer equals signalBased then the event referenced by ServiceEventDeployment.event shall be handled using signal-service-translation.⌋ *(RS_MANI_00063)*

Figure 11.4 illustrates a use-case where a ServiceInterface has two SomeipServiceInterfaceDeployments, one with signalBased serializer technology and one with someip.

The translation use-case is defined by having one RPortPrototype receiving the signalBased serialized messages and one PPortPrototype providing the someip serialized service.

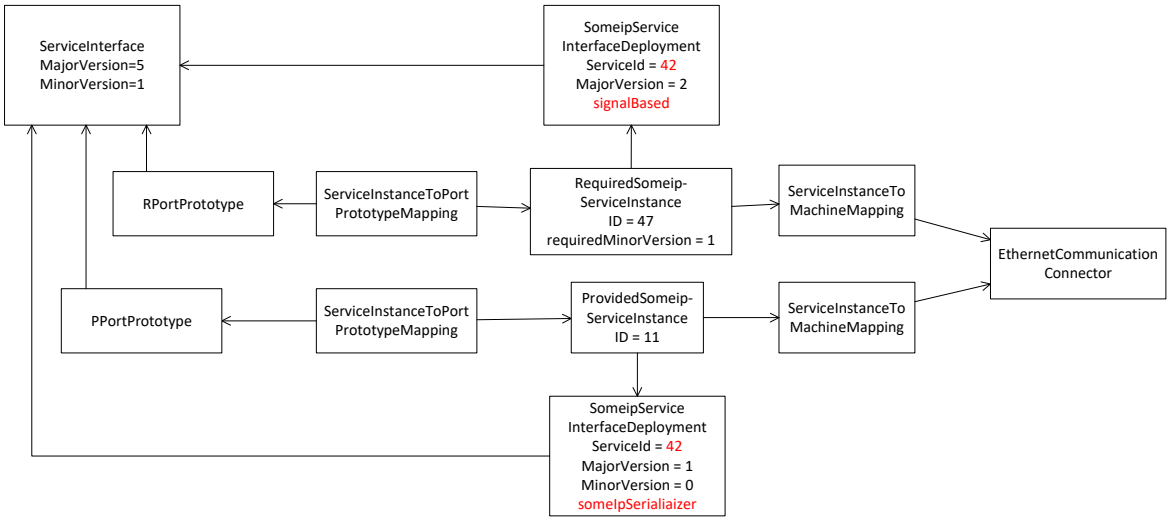


Figure 11.4: Example of 1 `PPortPrototype` mapped to one VLAN

11.4 Signal-To-Service Mapping

This chapter describes the mapping of `ServiceInterface` elements of a specific `AdaptivePlatformServiceInstance` to `ISignalTriggering`s.

Note that according to [TPS_MANI_03555] the same Ethernet socket (via `ISignal-Triggering`, `PduTriggering`, and `SocketConnection`) may be used for signal-based and service-oriented communication at the same time.

This allows to define one service instance which consists of events with different serialization technologies (i.e. `someip` and `signalBased`).

[TPS_MANI_03627]{DRAFT} **No signal-service-translation for methods** ⌈As Methods and Field getter/setter are already serialized using the `someip` serialization there is no need for signal-based translation for them.⌋*(RS_MANI_00029)*

Therefore, Methods and Field getter/setter are directly accessible using ara::com via the *Signal Based Network Binding* as defined in Communication Management [8]. For passing methods through refer to section 11.7.

The definition of the translation mappings is not directed (has no source or target point of view). Each individual mapping (`AbstractSignalBasedToISignalTrigger-ingMapping`) takes one element from a `ServiceInterface` and one `ISignal-Triggering`. The translation direction is determined by

- the `communicationDirection` of the referenced `ISignalPort` which is referenced by the `ISignalTriggering`

- the kind of service instance which is referenced from the ServiceInstanceToSignalMapping in the role serviceInstance. It can be either a ProvidedApServiceInstance or a RequiredApServiceInstance.

Class	ServiceInstanceToSignalMappingSet			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class represents a list of mappings of ServiceInstances to ISignalTriggerings. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInstanceToSignalMappingSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
serviceInstance ToSignal Mapping	ServiceInstanceTo SignalMapping	*	aggr	This is one particular mapping accociation of a Service Instance to a number of ISignalTriggerings, **Tags:**atp.Status=draft

Table 11.1: ServiceInstanceToSignalMappingSet

Class	ServiceInstanceToSignalMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class is defined for a specific ServiceInstance and contains the mappings of elements of a ServiceInterface for which the ServiceInstance is defined to individual ISignalTriggerings. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
eventElement Mapping	SignalBasedEvent ElementToISignal TriggeringMapping	*	aggr	Mapping of an event or an element inside of the event to an ISignalTriggering. **Tags:**atp.Status=draft
fieldMapping	SignalBasedFieldToI SignalTriggering Mapping	*	aggr	Mapping of a field to ISignalTriggerings. **Tags:**atp.Status=draft
methodMapping	SignalBasedMethodToI SignalTriggering Mapping	0..1	aggr	Mapping of a method to ISignalTriggerings. **Tags:**atp.Status=draft
serviceInstance	AdaptivePlatform ServiceInstance	0..1	ref	Reference to a ServiceInstance from which the corresponding ServiceInterface elements will be transported in the signal-based way over a communication medium. **Tags:**atp.Status=draft

Table 11.2: ServiceInstanceToSignalMapping

The ServiceInstanceToSignalMapping refers to an AdaptivePlatformServiceInstance and thereby defines which serviceInterfaceDeployment elements will be mapped by the aggregated eventElementMapping, methodMapping and/or fieldMapping to ISignalTriggerings. This is described in detail in the following chapters.

The ServiceInstanceToMachineMapping which refers to the AdaptivePlatformServiceInstance (which in turn is referenced by the ServiceInstanceToSignalMapping) defines on which CommunicationConnector (i.e. network / VLAN) the signal based communication shall be performed.

[TPS_MANI_03629]{DRAFT} **Relation of `ServiceInstanceToSignalMapping` and `CommunicationConnector`** ⌈The `ServiceInstanceToMachineMapping` referring to the `AdaptivePlatformServiceInstance` defines on which `CommunicationConnector` the `AdaptivePlatformServiceInstance` shall be communicated. If a `ServiceInstanceToSignalMapping` refers to the same `AdaptivePlatformServiceInstance` then the signal based communication shall be performed on the referenced `CommunicationConnector`.⌋*(RS_MANI_00029)*

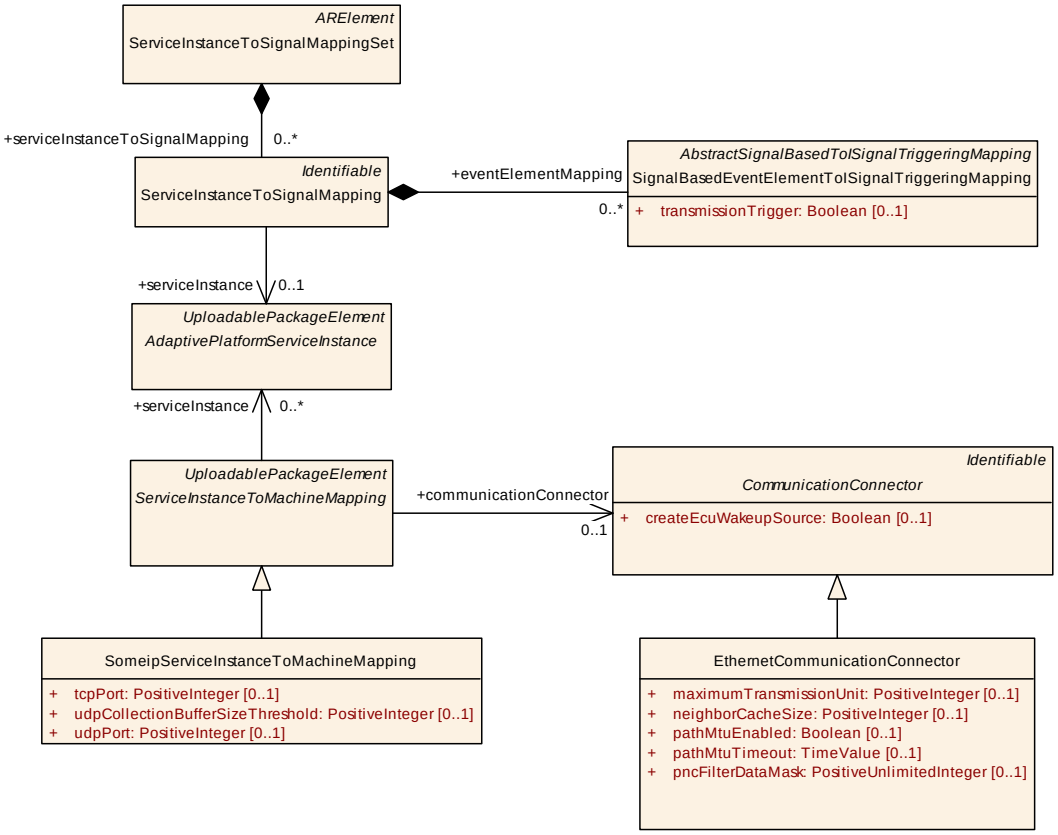


Figure 11.5: Relation of `ServiceInstanceToSignalMapping` and `CommunicationConnector`

11.4.1 SignalBasedEvent Mapping

It is required that every `event` with `signalBased` serialization has a `ServiceInstanceToSignalMapping` defined:

[constr_3550]{DRAFT} **Existence of `ServiceInstanceToSignalMapping` for an `event` with `signalBased` serialization** ⌈If an `event` is referenced by a `SomeipEventDeployment` in the role `event` and the attribute `SomeipEventDeployment.serializer` is set to `signalBased` then a `ServiceInstanceToSignalMapping` shall exist with `eventElementMapping` referring to the `event` in the role `dataPrototypeInServiceInterfaceRef`.⌋*()*

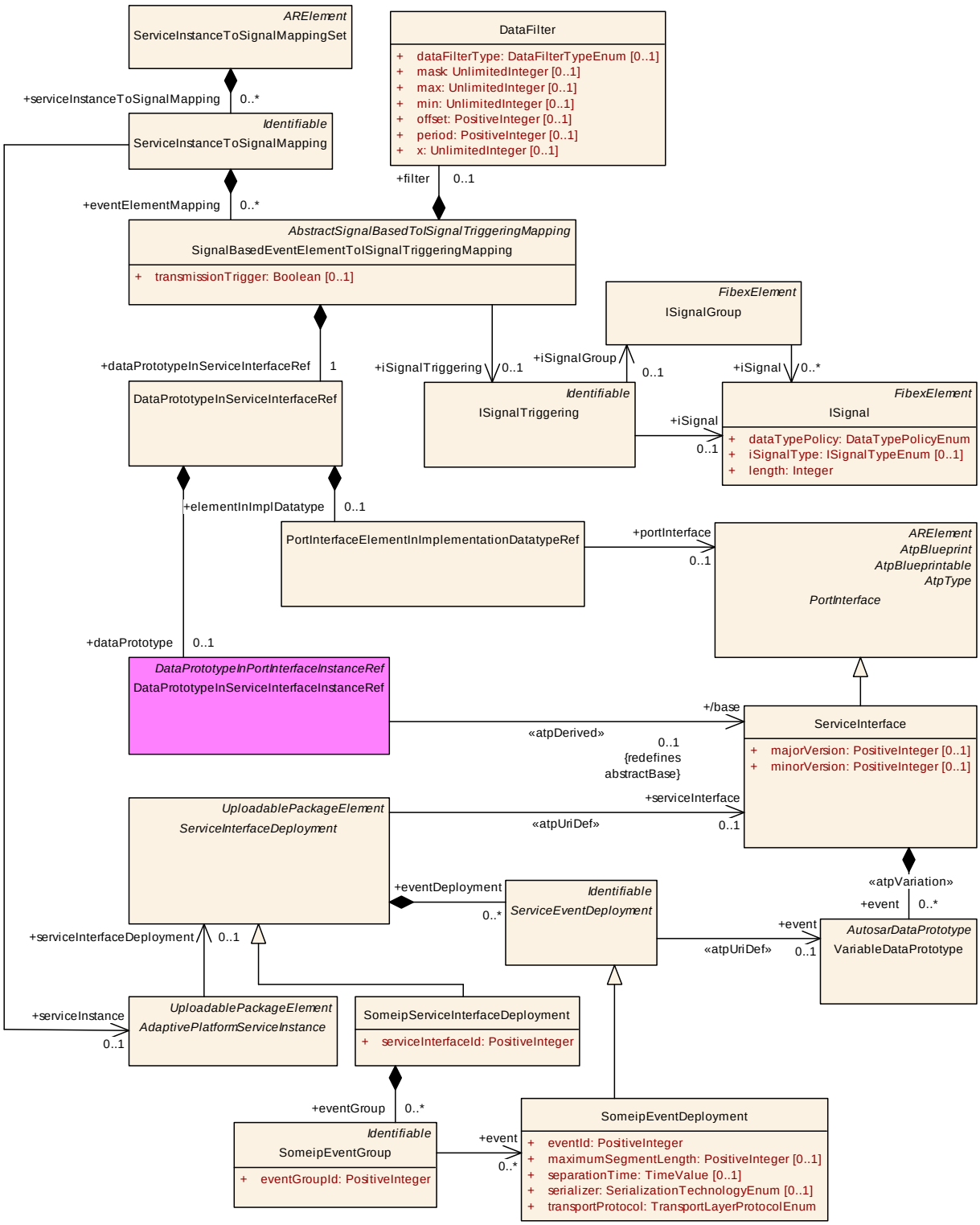


Figure 11.6: Mapping of Event elements to ISignals

In case of composite payload it is important that the mapping definition is complete from the target point of view:

[constr_3551]{DRAFT} **Full mapping of target `ISignalGroup`** ⌈If an `ISignalTriggering` is part of a `ServiceInstanceToSignalMapping` and the `ISignalTriggering` refers to an `ISignalPort` with `communicationDirection` equals `out` and the `ISignalTriggering` refers to an `ISignalGroup` in the role `iSignalGroup` then a `SignalBasedEventElementToISignalTriggeringMapping` shall exist for every `ISignal` referenced by the `ISignalGroup` in the role `iSignal`.⌋*()*

[constr_3552]{DRAFT} **Full mapping of target `event`** ⌈If the `ServiceInstanceToSignalMapping` refers to a `ProvidedSomeipServiceInstance` and the `dataPrototypeInServiceInterfaceRef` refers to a `DataPrototype` which is part of a composite data type then a `SignalBasedEventElementToISignalTriggeringMapping` shall exist for every `DataPrototype` that is part of the composite data type.⌋*()*

[TPS_MANI_03124]{DRAFT} **`ServiceInterface.event` to `ISignalTriggering` mapping** ⌈The `SignalBasedEventElementToISignalTriggeringMapping` meta-class provides the ability to map a `DataPrototype` defined in the context of a `ServiceInterface` to one `ISignalTriggering` of the `ISignal` or `ISignalGroup`.⌋*(RS_MANI_00029)*

Class	**SignalBasedEventElementToISignalTriggeringMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class defines the mapping of a ServiceInterface event or an element that is defined inside of the event in case that the datatype is composite to an ISignalTriggering. **Tags:**atp.Status=draft			
Base	*ARObject*, *AbstractSignalBasedToISignalTriggeringMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataPrototypeIn ServiceInterface Ref	DataPrototypeInService InterfaceRef	1	aggr	Reference to a DataPrototype or to an internal structure of a DataPrototype in the context of a ServiceInterface. **Tags:**atp.Status=draft
filter	DataFilter	0..1	aggr	Defines an optional filter to be applied during translation. **Tags:**atp.Status=draft
iSignal Triggering	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport a piece of data of an event that is defined in a ServiceInterface in a signal-based way over a communication channel. **Tags:**atp.Status=draft
transmission Trigger	Boolean	0..1	attr	Defines whether the source element triggers the sending of the respective payload.

Table 11.3: SignalBasedEventElementToISignalTriggeringMapping

In the example sketched in Figure 11.7 the *TestEvent* in the *TestServiceInterface* is of type *struct1* that consists of a primitive element *a* and struct *b*. The struct *b* consists of the primitive elements *x*, *y* and *z*.

One `ServiceInstanceToSignalMapping` with several `SignalBasedEventElementToISignalTriggeringMapping`s is necessary to map the *TestEvent* to the corresponding `ISignalTriggering`s.

One SignalBasedEventElementToISignalTriggeringMapping maps the DataPrototype that represents the *TestEvent* to an ISignalTriggering of an ISignalGroup. Here the dataPrototype role of DataPrototypeInServiceInterfaceRef is used to refer to the targetDataPrototype according to the rules defined in [TPS_MANI_01136] and [TPS_MANI_01137].

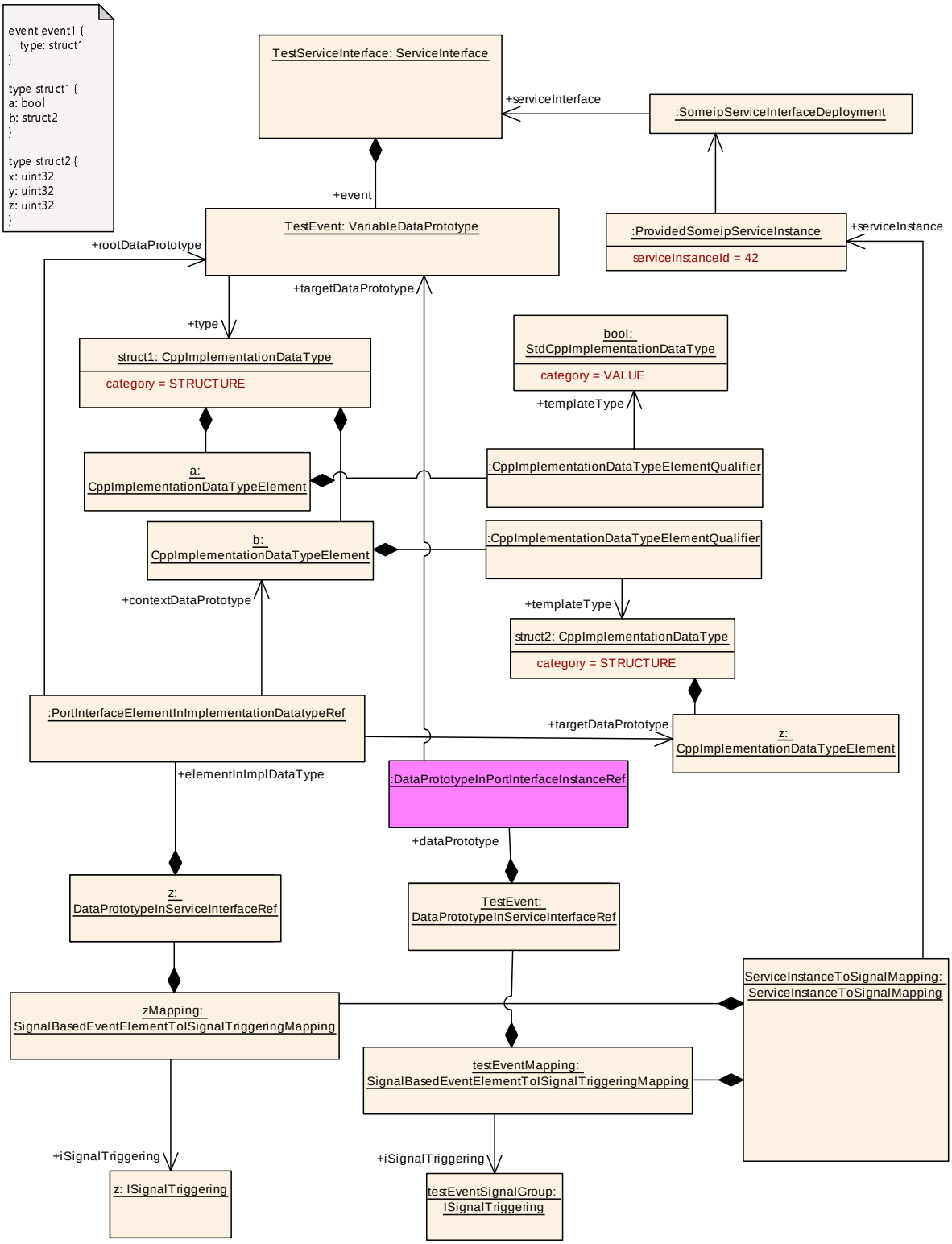


Figure 11.7: Example for a mapping of event content to a Signal

Additional SignalBasedEventElementToISignalTriggeringMappings are necessary to map the primitive DataPrototypes *a*, *x*, *y* and *z* to ISignalTriggerings of ISignals located in the ISignalGroup.

The example shows the mapping of *z* to the ISignalTriggering. Here the elementInImplDatatype role of DataPrototypeInServiceInterfaceRef is used to refer to the targetDataPrototype since it refers to the internal structure of an AutosarDataPrototype which is typed by a CppImplementationDataType. The context of this reference is defined by the *TestEvent* and struct *b*.

11.4.2 SignalBasedField Mapping

[TPS_MANI_03126]{DRAFT} **ServiceInterface.field mapping to ISignalTriggerings** ⌈The SignalBasedFieldToISignalTriggeringMapping metaclass provides the ability to map a field

- to one ISignalTriggering for the ISignalGroup representing the Notifier or
- to one ISignalTriggering for the ISignal representing the primitive Notifier element or
- to one ISignalTriggering for the ISignal representing the Getter-Call and one ISignalTriggering for the ISignal representing the Getter-Return or
- to one ISignalTriggering for the ISignal representing the Setter-Call and one ISignalTriggering for the ISignal representing the Setter-Return.

⌋*(RS_MANI_00029)*

It means that several SignalBasedFieldToISignalTriggeringMappings may be necessary to map a field to the corresponding ISignalTriggerings.

It is required that every field using a SomeipFieldDeployment.notifier with signalBased serialization has a ServiceInstanceToSignalMapping defined:

[constr_3553]{DRAFT} **Existence of ServiceInstanceToSignalMapping for an field with signalBased serialization** ⌈If a field is referenced by a SomeipFieldDeployment in the role field and that SomeipFieldDeployment aggregates a SomeipEventDeployment in the role notifier and the SomeipEventDeployment has an attribute SomeipEventDeployment.serializer set to signalBased then there shall exist a ServiceInstanceToSignalMapping with a fieldMapping referring to the field in the role dataPrototypeInServiceInterfaceRef and the SignalBasedFieldToISignalTriggeringMapping shall refer to a ISignalTriggering in the role notifierSignalTriggering.⌋*()*

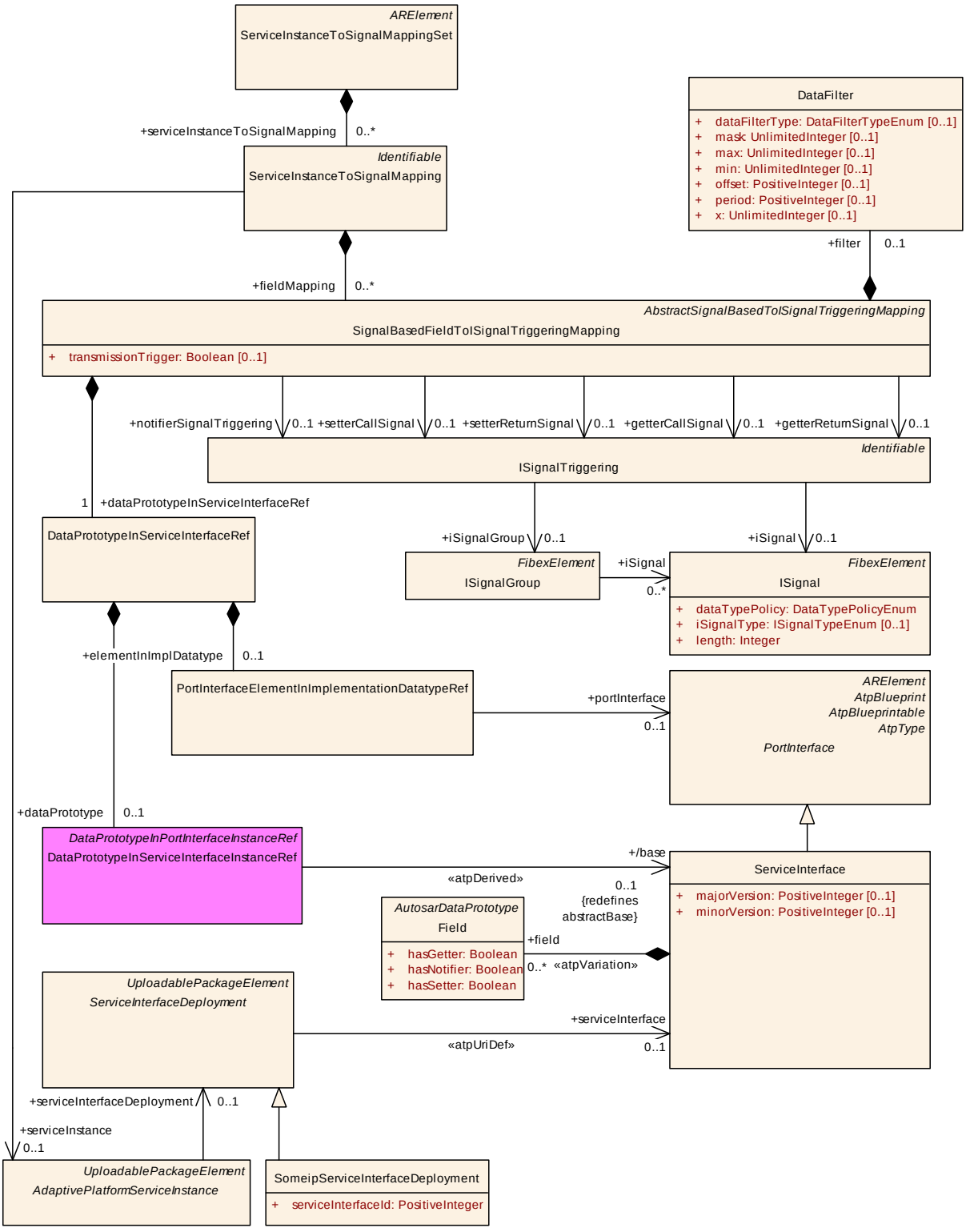


Figure 11.8: Mapping of Fields to ISignals

In the example sketched in Figure 11.9 the *testField* in the *testServiceInterface* is of type *struct1* that consists of the primitive elements *a* and *b*. The *testField* defines a notifier and a setter method.

Class	**SignalBasedFieldToISignalTriggeringMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class defines the mapping of a ServiceInterface field to ISignalTriggerings that represent the notifier elements, the getter call and response, the setter call and response on a signal-based communication channel. . **Tags:**atp.Status=draft			
Base	*ARObject*, *AbstractSignalBasedToISignalTriggeringMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataPrototypeIn ServiceInterface Ref	DataPrototypeInService InterfaceRef	1	aggr	Reference to a DataPrototype or to an internal structure of a DataPrototype in the context of a ServiceInterface. **Tags:**atp.Status=draft
filter	DataFilter	0..1	aggr	Defines an optional filter to be applied during translation. **Tags:**atp.Status=draft
getterCallSignal	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport the getter method call in a signal-based way over a communication channel. **Tags:**atp.Status=draft
getterReturn Signal	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport the getter method response in a signal-based way over a communication channel. **Tags:**atp.Status=draft
notifierSignal Triggering	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport a piece of data of a notifier in a signal-based way over a communication channel. **Tags:**atp.Status=draft
setterCallSignal	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport the setter method call in a signal-based way over a communication channel. **Tags:**atp.Status=draft
setterReturn Signal	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport the setter method response in a signal-based way over a communication channel. **Tags:**atp.Status=draft
transmission Trigger	Boolean	0..1	attr	Defines whether the source notifier element triggers the sending of the respective payload.

Table 11.4: SignalBasedFieldToISignalTriggeringMapping

One `SignalBasedFieldToISignalTriggeringMapping` maps the *TestField* to `ISignalTriggering`s for the Setter-Call and Setter-Return. Here the `dataPrototype` role of `DataPrototypeInServiceInterfaceRef` is used to refer to the `targetDataPrototype` (`field`) according to the rules defined in [TPS_MANI_01136] and [TPS_MANI_01137].

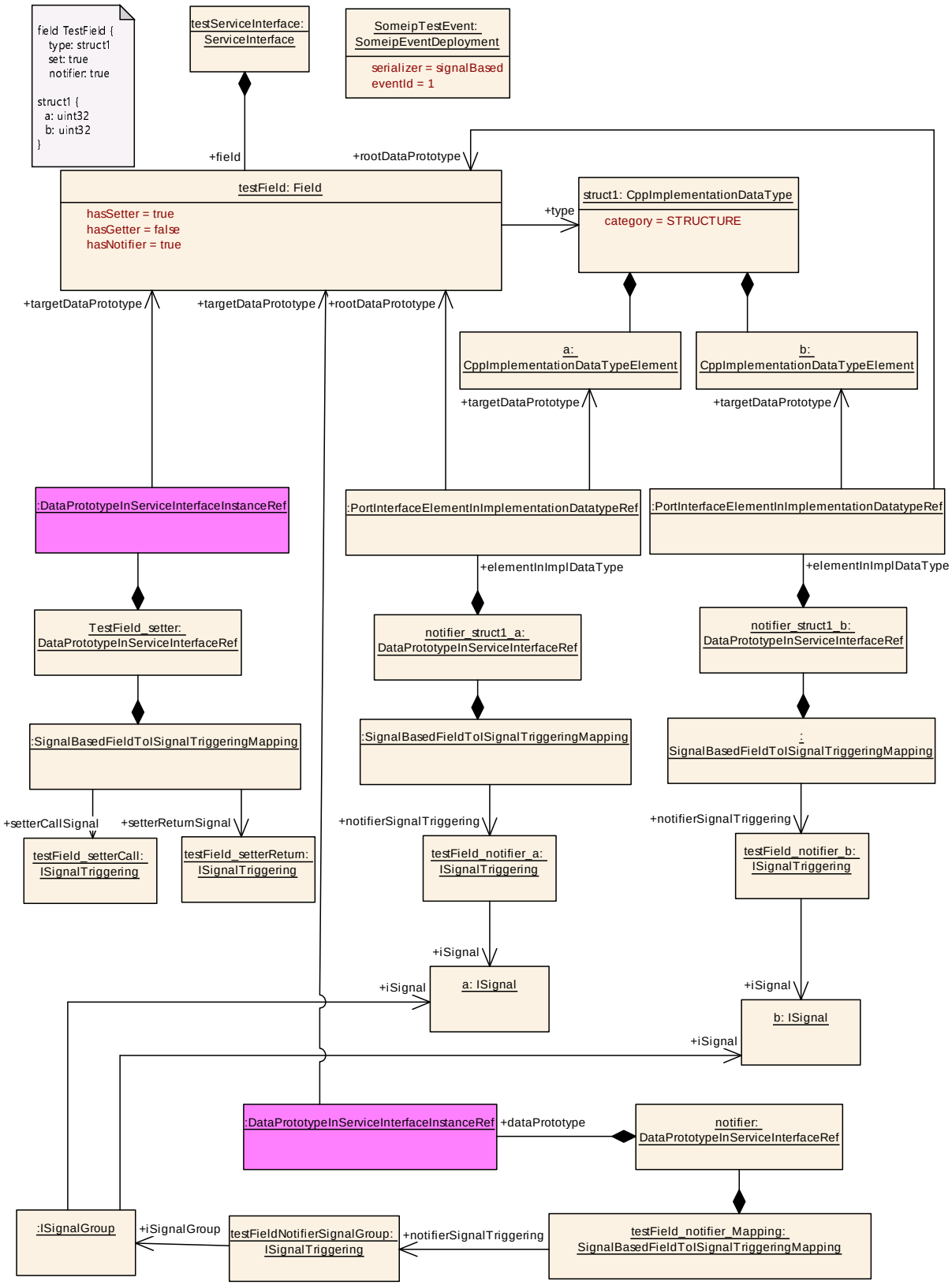


Figure 11.9: Example for a mapping of a field to Signals

Additional SignalBasedFieldToISignalTriggeringMappings are necessary to map the field notifier to the corresponding ISignalTriggerings. One SignalBasedFieldToISignalTriggeringMapping maps the *TestField* to the ISignalTriggering of the ISignalGroup that collects all ISignals that transport the content of the notifier.

The primitive DataPrototypes *a* and *b* are mapped by additional SignalBasedFieldToISignalTriggeringMappings to ISignalTriggerings of ISignals located in the ISignalGroup. Here the elementInImplDatatype role of DataPrototypeInServiceInterfaceRef is used to refer to the targetDataPrototype since it refers to the internal structure of an AutosarDataPrototype which is typed by a CppImplementationDataType. The context of this reference is defined by the *testField*.

11.4.3 SignalBasedMethod Mapping

[TPS_MANI_03125]{DRAFT} **ServiceInterface.method to ISignalTriggerings mapping** ⌈The SignalBasedMethodToISignalTriggeringMapping meta-class provides the ability to map a method to one ISignalTriggering for the ISignal representing the Method-Call and one ISignalTriggering for the ISignal representing the Method-Return.⌋ *(RS_MANI_00029)*

Class	**SignalBasedMethodToISignalTriggeringMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class defines the mapping of a ServiceInterface method to a ISignalTriggering. **Tags:**atp.Status=draft			
Base	*ARObject*, *AbstractSignalBasedToISignalTriggeringMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
callSignal Triggering	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport the method call in a signal-based way over a communication channel. **Tags:**atp.Status=draft
method	ClientServerOperation	1	ref	Reference to a method defined in the context of a Service Interface. **Tags:**atp.Status=draft
returnSignal Triggering	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport the method response in a signal-based way over a communication channel. **Tags:**atp.Status=draft

Table 11.5: SignalBasedMethodToISignalTriggeringMapping

Please note that the SignalBasedMethodToISignalTriggeringMapping shall also be used for the mapping of methods where the value of attribute method.fireAndForget is set to true. In this case, only the callSignalTriggering shall be used since in the fire and forget Message Exchange Pattern only one message is sent from the service consumer to the service provider.

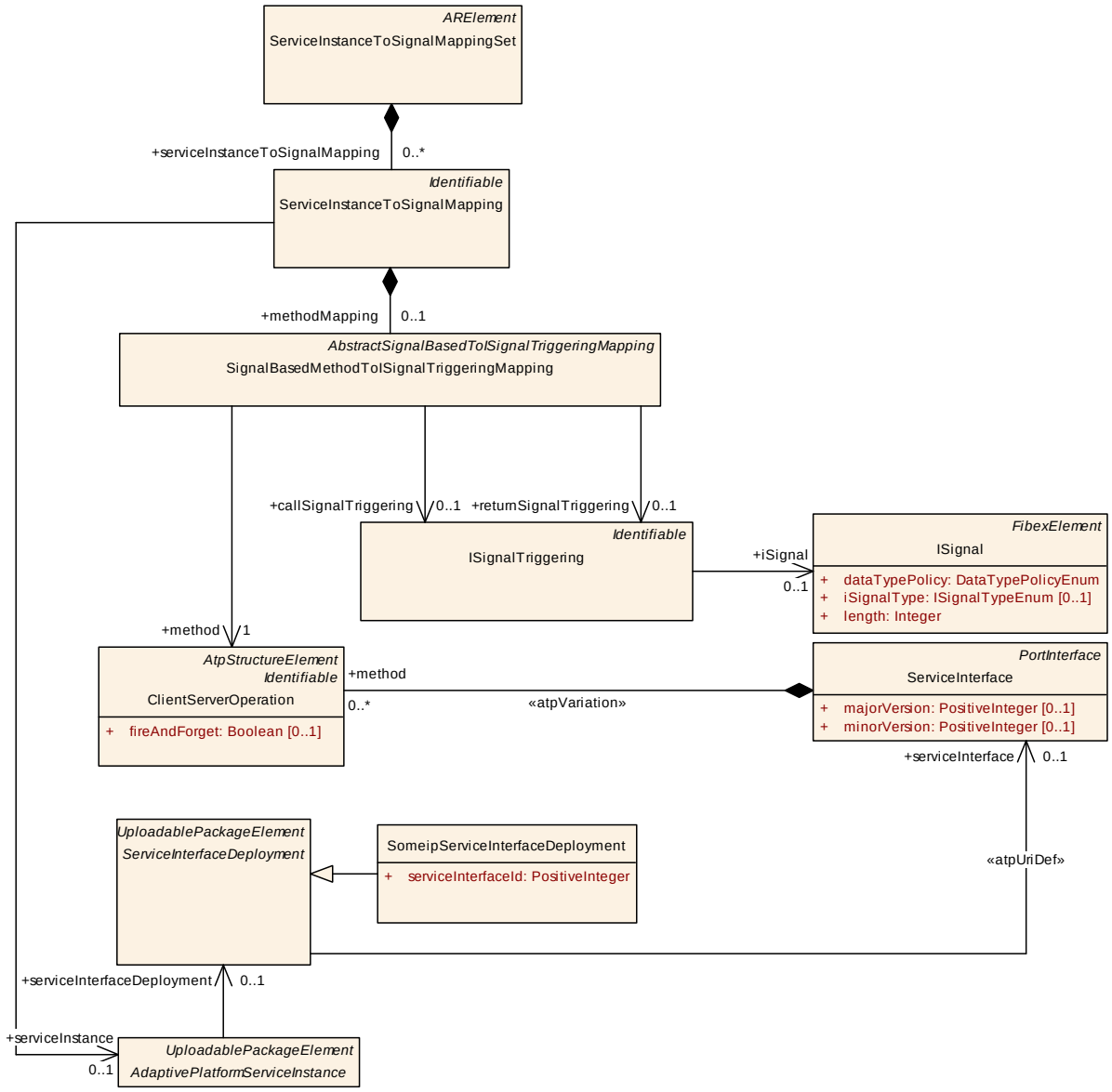


Figure 11.10: Mapping of Methods to ISignals

In the example sketched in Figure 11.11 the *Calibrate* method in the *TestServiceInterface2* is mapped with a single `SignalBasedMethodToISignalTriggeringMapping` to `ISignalTriggering`s for the Call and Return.

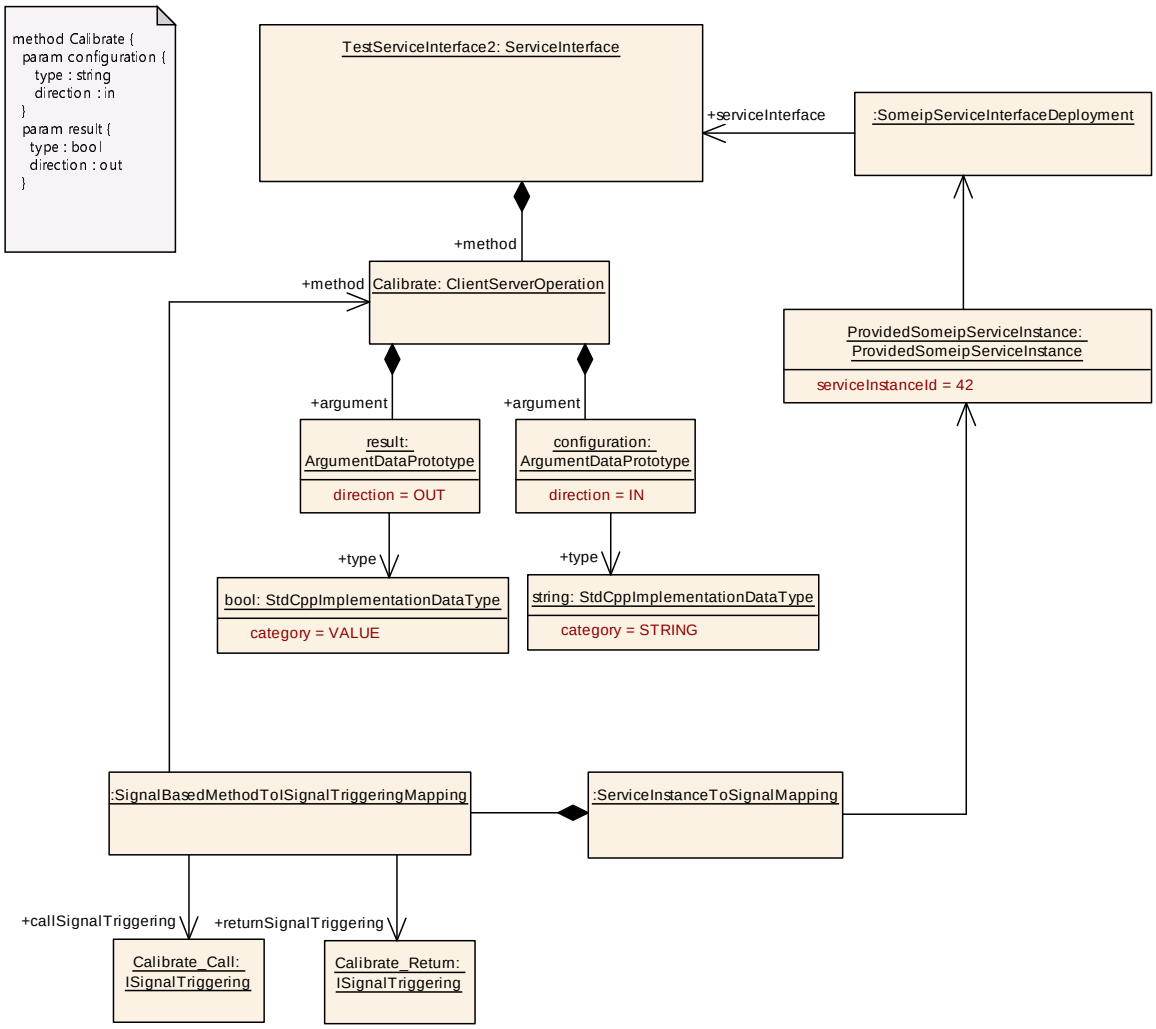


Figure 11.11: Example for a mapping of a method to Signals

11.5 Service discovery control

The signal-service-translation needs to determine when the translated service (service which originates in signal-based messages) shall actually be offered/subscribed.

Attributes defining the behavior of signal-service-translation are available at the `SignalServiceTranslationProps`. The reference `serviceElementMapping` determines to which service instance these settings apply.

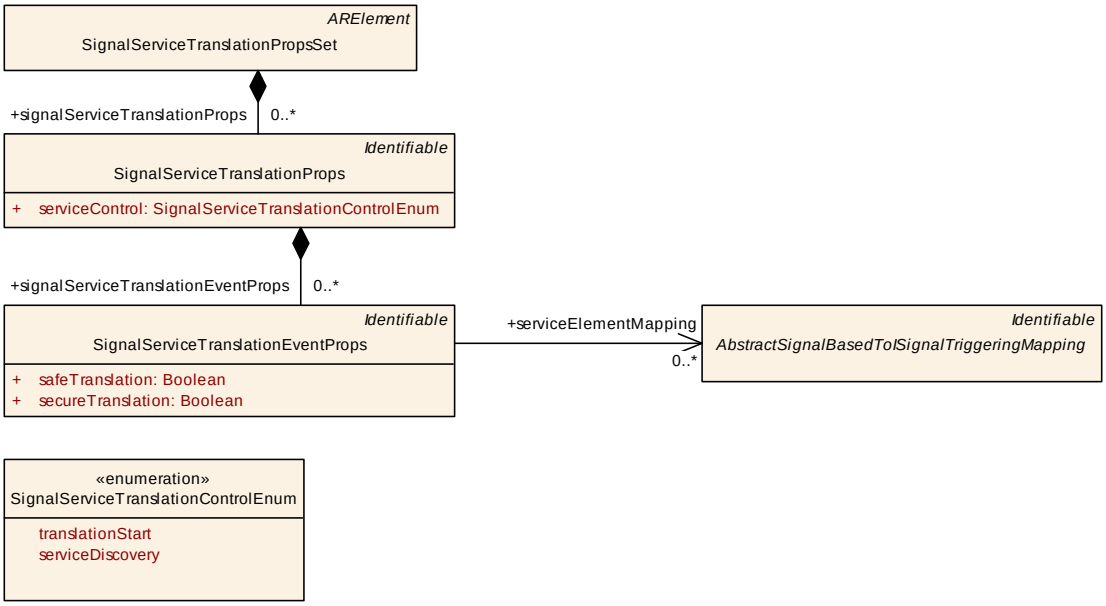


Figure 11.12: Signal Service Translation properties

Class	**SignalServiceTranslationPropsSet**			
Package	M2::AUTOSARTemplates::CommonStructure::SignalServiceTranslation			
Note	Collection of SignalServiceTranslationProps. **Tags:**atp.recommendedPackage=SignalServiceTranslationProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
signalService Translation Props	SignalService TranslationProps	*	aggr	Collection of SignalServiceTranslationProps.

Table 11.6: SignalServiceTranslationPropsSet

Class	**SignalServiceTranslationProps**			
Package	M2::AUTOSARTemplates::CommonStructure::SignalServiceTranslation			
Note	This element allows to define the properties which are applicable for the signal-service-translation service.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
serviceControl	SignalService TranslationControlEnum	1	attr	Defines how the service instance control shall behave.
signalService Translation EventProps	SignalService TranslationEventProps	*	aggr	Defines properties for a single translated event.

Table 11.7: SignalServiceTranslationProps

Class	**SignalServiceTranslationEventProps**			
Package	M2::AUTOSARTemplates::CommonStructure::SignalServiceTranslation			
Note	This element allows to define the properties which are applicable for the signal-service-translation event.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
safeTranslation	Boolean	1	attr	Defined whether the translation shall happen in a safe way.
secure Translation	Boolean	1	attr	Defined whether the translation shall happen in a secure way.
serviceElement Mapping	AbstractSignalBasedToI SignalTriggering Mapping	*	ref	Reference to the collection of SignalBased to ISignal Triggerung mappings the properties apply to. **Tags:**atp.Status=draft

Table 11.8: SignalServiceTranslationEventProps

At which point in time a specific service instance (originating from signal service translation) is actually offered / subscribed at the service discovery can be defined per service instance:

Possible approaches for service availability/subscription are:

- `translationStart` - right after translation software is started
- `serviceDiscovery` - availability of related service instance

The attribute `SignalServiceTranslationProps.serviceControl` defines the service instance control behavior.

Enumeration	**SignalServiceTranslationControlEnum**
Package	M2::AUTOSARTemplates::CommonStructure::SignalServiceTranslation
Note	This enumeration allows to define how the service instance offer/subscribe control shall behave.
Literal	***Description***
serviceDiscovery	Defines the start of service control when other service is available. **Tags:**atp.EnumerationLiteralIndex=2
translationStart	Defines the start of service control at translation start. **Tags:**atp.EnumerationLiteralIndex=0

Table 11.9: SignalServiceTranslationControlEnum

11.5.1 Service control right after translation start

If the availability of the signal-based PDUs is not controlled then the respective translated services offers/subscriptions may be activated immediately at start of the translation software.

[TPS_MANI_03580]{DRAFT} **Service offer at startup** ⌈For a provided translated service instance, if the `SignalServiceTranslationProps.serviceControl` equals `translationStart` then the translation software shall - right after translation software start - *offer* the respective translated service instance.⌋ *(RS_MANI_00063)*

[TPS_MANI_03581]{DRAFT} **Service find at startup** ⌈For a required translated service instance, if the `SignalServiceTranslationProps.serviceControl` equals `translationStart` then the translation shall right after translation software start issue the *find* of the respective translated service instance and *subscribe* to its event groups.⌋*(RS_MANI_00063)*

11.5.2 Service control due to availability of related service instance

There are scenarios where the signal-based PDUs are actually controlled using SOME/IP Service Discovery. So there are services defined using events/methods/-fields and the service instances are offered / subscribed using the SOME/IP Service Discovery, just the payload of such services is not serialized according to the SOME/IP transformer rules (or a subset of events uses signal-based serialization). Therefore, a signal service translation is required for the payload.

11.5.2.1 Signal to Service

[TPS_MANI_03582]{DRAFT} **Service find for required signal** ⌈For a required translated service instance, if the `SignalServiceTranslationProps.serviceControl` equals `serviceDiscovery` then upon startup the translation software component shall issue a *service find* for the `RequiredSomeipServiceInstance` which is referenced by the `ServiceInstanceToSignalMapping` in the role `serviceInstance`.⌋*(RS_MANI_00063)*

[TPS_MANI_03583]{DRAFT} **Service subscribe for required signal** ⌈For a required translated service instance, if the `SignalServiceTranslationProps.serviceControl` equals `serviceDiscovery` and the *service find* of [TPS_MANI_03582] was successful then the translation software component shall issue a *subscribe* to all `SomeipServiceInterfaceDeployment.eventGroup`s.⌋*(RS_MANI_00063)*

11.5.2.2 Service to Signal

[TPS_MANI_03606]{DRAFT} **Service offer for provided signal** ⌈For a provided translated service instance, if the `SignalServiceTranslationProps.serviceControl` equals `serviceDiscovery` then upon startup the translation software component shall issue an *offer* for the `ProvidedSomeipServiceInstance` which is referenced by the `ServiceInstanceToSignalMapping` in the role `serviceInstance`.⌋*(RS_MANI_00063)*

11.6 Translation behavior

The signal-service-translation is defined as a network binding for the Communication Management [8] and it is not specified at which exact point in time or in which context the data transformation (i.e. signal-service-translation) will be executed.

The behavior of signal-service-translation is governed by the Adaptive Application which calls `ara::com`-APIs to interact with the communication management.

Data filtering:

If there is a `filter` defined at the `NonqueuedReceiverComSpec` then the evaluation of this `DataFilter` is performed in the COM-Stack. Thus, the COM-Stack filtering usually can not be applied when there are transformers involved because the state machines of E2E transformers need to receive every message.

The `NonqueuedReceiverComSpec.filter` only applies to the usage in the context of a signal-service-translation.

[constr_3562]{DRAFT} **Existence of `NonqueuedReceiverComSpec.filter`** ⌈The attribute `NonqueuedReceiverComSpec.filter` shall only exist if the referenced `dataElement` refers to an `AutosarDataPrototype` which is referenced by either a `SignalBasedEventElementToISignalTriggeringMapping` or a `SignalBasedFieldToISignalTriggeringMapping`.⌋*()*

If a data filtering shall be applied *after* the data transformation inside the signal-service-translation then there is the possibility to define a `DataFilter` at the `SignalBasedEventElementToISignalTriggeringMapping` and `SignalBasedFieldToISignalTriggeringMapping` in the role `filter`.

[TPS_MANI_03621]{DRAFT} **Data filter inside the signal-service-translation** ⌈If there is a `SignalBasedEventElementToISignalTriggeringMapping.filter` (resp. `SignalBasedFieldToISignalTriggeringMapping.filter`) defined this filtering shall be implemented after the COM-Stack and transformer parts have been processed.⌋*(RS_MANI_00063)*

Class	**NonqueuedReceiverComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes specific to non-queued receiving.			
Base	*ARObject*, *RPortComSpec*, *ReceiverComSpec*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
filter	DataFilter	0..1	aggr	The applicable filter algorithm for filtering the value of the corresponding dataElement.

Table 11.10: NonqueuedReceiverComSpec

11.6.1 Translation from one source

The simplest translation approach is a one source translation. If the translation target is primitive then it has by nature only one source. If the translation target is a composite data then it depends on the structure of the sources whether a one source translation is possible.

[TPS_MANI_03620]{DRAFT} **Service discovery control** ⌈If the service discovery control is enabled for a specific service instance then any payload coming / going to that service instance shall be translated and forwarded to the mapped output(s).⌋ *(RS_MANI_00063)*

Every time an input signal or event arrives it is translated to the mapped output path. Since this is a one source translation the arrival of the input signal or event is the trigger for the translation. Therefore, if all the translation mappings have been performed the target payload can be sent out.

The example in figure 11.13 shows several mapping options:

On the left side the straight forward approach is illustrated (one-to-one mapping) where the source `ISignal` structure and the `ServiceInterface` definition match and the mapping follows the structure (here the translation is signal to service).

On the right side a partial mapping is shown where each target `ISignal` or `ISignalGroup` is composed out of one source (here the translation is service to signal). Of course all target elements are mapped, however there may exist source elements which are not mapped (e.g. *z*).

Also, a split of information is shown where the content of *event x* is split: *a* and *b* are mapped to *x1* while *c* and *d* are mapped to *x2*.

A fan-out is also possible as shown with the input *d* which is mapped to *h* as well as to *j*.

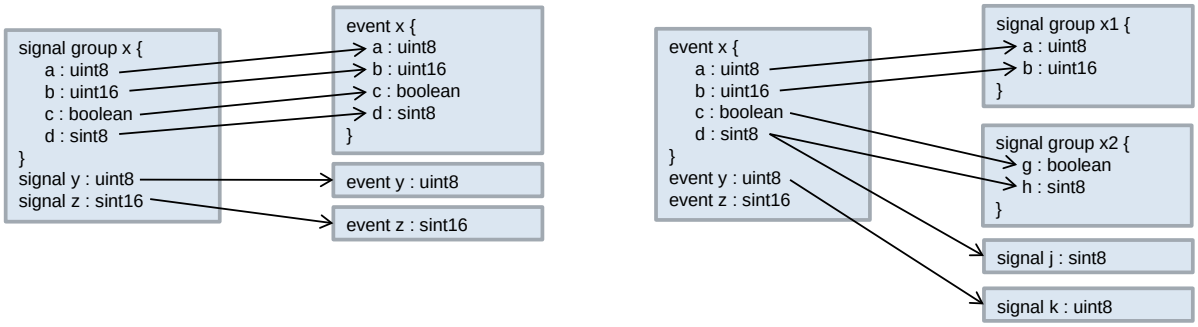


Figure 11.13: Example translation mapping from one source

11.6.2 Translation from several sources

There is also the use-case to support the composition of data from different sources. Because the sources will arrive in a not coordinated way, it is essential to define which part of the source mapped data actually triggers the sending of the target payload.

[TPS_MANI_03584]{DRAFT} **Definition of transmission triggers for translations with different sources** ⌈The attribute `SignalBasedEventElementToISignalTriggeringMapping.transmissionTrigger` respectively `SignalBasedFieldToISignalTriggeringMapping.transmissionTrigger` defines which translation parts contribute to the transmission triggering for the mapped payload.⌋*(RS_MANI_00063)*

[TPS_MANI_03588]{DRAFT} **Full translation before transmission triggering** ⌈In case there has been a transmission trigger caused by a source signal the signal-service-translation shall process all other mapped source signals from the triggering source context (signal group or IPdu) before actually sending out the target.⌋*(RS_MANI_00063)*

[TPS_MANI_03587]{DRAFT} **Transmission trigger for translations with different sources** ⌈If the attribute `SignalBasedEventElementToISignalTriggeringMapping.transmissionTrigger` respectively `SignalBasedFieldToISignalTriggeringMapping.transmissionTrigger` equals *true* then the reception of the respective source signal does cause the sending of the target (after all mapped sources from the same source context have been translated, see [TPS_MANI_03588]).⌋*(RS_MANI_00063)*

[TPS_MANI_03586]{DRAFT} **No transmission trigger for translations with different sources** ⌈If the attribute `SignalBasedEventElementToISignalTriggeringMapping.transmissionTrigger` respectively `SignalBasedFieldToISignalTriggeringMapping.transmissionTrigger` is not defined or has the value *false* then the reception of the respective source signal does not cause the sending of the target.⌋*(RS_MANI_00063)*

The example in figure 11.14 shows the case where for the composite mapping target the sources are located in different structures. The content of the event *x1* is composed out of parts from signal group *x* as well as from signal *y*. The attribute `transmissionTrigger` = *true* defines which sources trigger the sending of the target payload.

The source signal *x.a* as well as source signal *y* are the trigger for the translation of event *x1*. This means that upon reception of signal group *x* the source values of *x.a* and *x.b* are translated into the target *x1.a* and *x1.b* and the event *x1* is sent out. Also, when the signal *y* is received the source value of *y* is translated into the target *x1.y* and the event *x1* is sent out (again).

For target event *x2* the source signal *z* is defined as the only trigger for the translation. This means that upon reception of signal *y* the source values of *y* is translated into the target *x2.g* but the event *x2* is not yet sent out.

Only when the triggering source signal *z* is received the value of signal *z* is translated into the target *s2.h* and the event *x2* is sent out.

This also means that for several receptions of source signal *y* only the latest translated values of *x2.g* is actually sent out (when the triggering source signal *z* is received).

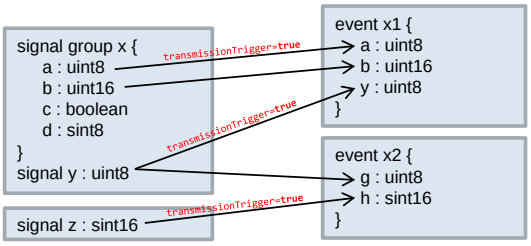


Figure 11.14: Example translation mapping from several sources

11.7 Translation pass-through composition

In case a *Pass-through Composition* is defined (see figure 11.2) the behavior for the defined pass-through definition shall be implemented. It is well possible to automatically generate the implementation of such a pass-through behavior out of the information given in the *Pass-through Composition*.

The mapping approach is already used in section 3.5 where the `PassThroughSwConnector` is used to define the matching `PortPrototype`s for the *facade* use-case. Specifically [constr_5056] applies as well for the signal-service-translation.

SOME/IP services may consist of a mixture of events, fields and methods. While events and field notifiers are (potentially) subject to signal-service-translation it is well-defined that methods (methods, getter- and setter-methods, and fire-and-forget methods) are serialized according to the SOME/IP serialization rules.

However, the mixed nature of a specific service instance makes it necessary that methods also have to be considered in the *Pass-through Composition*. It is not so much about the translation of the payload serialization, but the general wrapping of events, fields and methods in one service instance makes it necessary that methods get *passed through* as well.

11.8 Expected features of Classic platform

Classic AUTOSAR SWS-COM Specification provides a variety of possibilities to pack signals into PDUs - from a structural as well as from a behavioral perspective. Also, the further layers in the COM-Stack may influence the PDUs layout.

The goal of this section is to specify which of these Classic COM-Stack features shall also be available on the Adaptive Platform signal-service-translation.

11.8.1 Processing order

For the features of the Classic platform COM-Stack there is a well-defined processing order in which the actions are performed upon Tx and Rx of data. It is required for the signal-service-translation to ensure the same processing order.

[TPS_MANI_03585]{DRAFT} **Processing order of COM-Stack features** ⌈For the COM-Stack features the signal-service-translation supports the processing order shall be the same as on the Classic platform.⌋*(RS_MANI_00063)*

11.8.2 Reception data filter

[TPS_MANI_03589]{DRAFT} **Reception data filter of COM-Stack** ⌈If the `ISignalTriggering` refers to an `ISignalPort` with `communicationDirection` equal `in` and the `ISignalPort` has a `dataFilter` defined then the signal-service-translation shall perform filtering of the respective payload data according to the Classic platform *Com* module specification [30].

The following filter settings are supported by signal-service-translation:

- `always`
- `never`
- `maskedNewEqualsX`
- `maskedNewDiffersX`
- `maskedNewDiffersMaskedOld`

⌋*(RS_MANI_00063)*

Note that the setting of `maskedNewDiffersMaskedOld` requires the translation software to keep the previous (*old*) value for comparison.

Also, note that data filtering may spoil E2E reception since it filters out receptions and influences the sequence counter checks.

11.8.3 Reception of invalid signal

[TPS_MANI_03592]{DRAFT} **`ISignal` invalidation of COM-Stack** ⌈If the `ISignalTriggering` refers to an `ISignalPort` with `communicationDirection` equal `in` then the following values for `ISignalPort.handleInvalid` shall be supported by the signal-service-translation:

- `dontInvalidate`
- `replace`

⌋*(RS_MANI_00063)*

[TPS_MANI_03593]{DRAFT} **handleInvalid = dontInvalidate behavior of COM-Stack** ⌈If the ISignalTriggering refers to an ISignalPort with communicationDirection equal in and the ISignalPort.handleInvalid equals dontInvalidate then the signal-service-translation shall not perform any invalidation handling.⌋*(RS_MANI_00063)*

[TPS_MANI_03594]{DRAFT} **handleInvalid = replace behavior of COM-Stack** ⌈If the ISignalTriggering refers to an ISignalPort with communicationDirection equal in and the ISignalPort.handleInvalid equals replace and the received value of the ISignal equals the ISignal.networkRepresentationProps.invalidValue then the signal-service-translation shall *replace* the invalidValue with the ISignal.initValue.⌋*(RS_MANI_00063)*

11.8.4 Update Bit handling

[TPS_MANI_03595]{DRAFT} **Update Bit support for ISignal** ⌈If the ISignalTriggering refers to an ISignalPort with communicationDirection equal in and the ISignalTriggering refers to an ISignal and that ISignal is mapped into an ISignalIPdu with a ISignalToIPduMapping.updateIndicationBitPosition defined then

- if the received *update bit* is *true* then the respective ISignal shall be considered for reception
- if the received *update bit* is *false* then the respective ISignal shall not be considered for reception.

⌋*(RS_MANI_00063)*

[TPS_MANI_03596]{DRAFT} **Update Bit support for ISignalIPdu** ⌈If the ISignalTriggering refers to an ISignalPort with communicationDirection equal in and the ISignalTriggering refers to an ISignal and that ISignal is mapped into an ISignalIPdu and the ISignalIPdu is mapped into a Frame with a PduToFrameMapping.updateIndicationBitPosition defined then

- if the received *update bit* is *true* then the respective ISignalIPdu shall be considered for reception
- if the received *update bit* is *false* then the respective ISignalIPdu shall not be considered for reception.

⌋*(RS_MANI_00063)*

11.8.5 Transfer properties and transmission modes for Service to Signal

[TPS_MANI_03590]{DRAFT} **Transfer properties and transmission modes of COM-Stack** ⌈If the `ISignalTriggering` refers to an `ISignalPort` with `communicationDirection` equal `out` and the `ISignalTriggering` refers to an `ISignal` and that `ISignal` is mapped into an `ISignalIPdu` with an `iPduTimingSpecification` and the `IPduTiming` has a `transmissionModeDeclaration` defined and the `TransmissionModeDeclaration` has a `transmissionModeCondition` defined and the following values for `TransmissionModeCondition.dataFilter` shall be supported by the signal-service-translation:

- `always`
- `never`
- `maskedNewEqualsX`
- `maskedNewDiffersX`

⌋*(RS_MANI_00063)*

11.8.6 Deadline monitoring

The Adaptive Application is responsible for the deadline monitoring.

11.8.7 Signal and IPdu Transmission

The Adaptive Application is responsible for the periodic production of data and triggering of sending.

11.8.8 IPdu multiplexing

On low-payload networks there is also the usage of IPdu Multiplexing to handle the limited number of identifiers for the respective network transport (e.g. CAN-Ids).

[TPS_MANI_03597]{DRAFT} **Support for `MultiplexedIPdu`** ⌈The signal-service-translation shall support the handling of `MultiplexedIPdu` defined payload. The support shall be available for sending and receiving of `MultiplexedIPdu`s.⌋*(RS_MANI_00063)*

11.9 End-to-End considerations

The Signal Service Translation on Adaptive platform follows a single-sided approach on ara::com. To the Adaptive Application there is no difference whether the service is

communicated using SOME/IP or DDS, or whether the service originates in a signal-service-translation. As the translation is running in the same `Process` as the adaptive application itself, the adaptive application has direct access to the translated service payload.

In case there is an E2E header attached and/or a secure communication defined the translation software needs to check the validity (for reception) or calculate the CRC/-MAC (for sending) of signal-based payload.

The information whether and which E2E / Security profile is configured is defined in the System Template [17] part belonging to the mapping information of sections 11.4.1, 11.4.2, and 11.4.3.

The assignment of the `ISignal`s to `ISignalIPdu`s which in turn are assigned to `SecuredIPdu`s defines the security aspects.

The assignment of `ISignal`s to `EndToEndTransformationDescription`s define the safety aspects.

11.9.1 Safety

The attribute `SignalServiceTranslationEventProps.safeTranslation` is used to explicitly require that the translation shall be configured in a safe transport and that the translation software shall handle the translation activity in an end-to-end preserving way.

[TPS_MANI_03607]{DRAFT} **Handling of safe signal-service-translation in one `Executable`** ⌈It is required that the signal-service-translation (and vice versa) of one Service/SignalGroup pair which are mapped to each other, shall be handled in one `Executable` to also cover a closed mapping from one E2E profile to another, if necessary.

The signal-service-translation of different (independent) Services/SignalGroups may be handled by different `Executable`s.⌋*(RS_MANI_00063)*

[TPS_MANI_03608]{DRAFT} **Support for safe signal-service-translation and service-signal-translation** ⌈The translation of E2E protected data shall be supported in both directions, signal-service-translation and service-signal-translation.⌋ *(RS_MANI_00063)*

[TPS_MANI_03609]{DRAFT} **Support for safe signal-service-translation with same or different E2E profiles** ⌈The translation of E2E protected data shall support the occurrence of

- the same E2E profile on both sides of the communication and
- different E2E profiles on each side of the communication.

⌋*(RS_MANI_00063)*

[TPS_MANI_03610]{DRAFT} **1:n mapping for E2E protected data** ⌈It shall be possible to map the same E2E protected source data to several E2E protected target data (1:n).⌋*(RS_MANI_00063)*

[TPS_MANI_03611]{DRAFT} **E2E protected target out of E2E protected sources** ⌈The content of one E2E protected target shall only be composed out of data from E2E protected sources.⌋*(RS_MANI_00063)*

The rationale for [TPS_MANI_03611] is to support the use-case where target data shall be E2E protected and it is composed of several sources.

[TPS_MANI_03614]{DRAFT} **No translation of not OK E2E protected data out of several sources** ⌈If a E2E protected source data is mapped into a composed E2E protected target data (according to [TPS_MANI_03611]) and
if the E2E-Check for the source data returns any E2E error (not *E_OK*) then
this source data shall not be forwarded to the respective target data and (if applicable) shall not trigger the transmission of the target.⌋*(RS_MANI_00063)*

If source data is not verified as *E_OK* it is not translated. If the translated E2E protected data comes from several sources there may occur correlation and synchronicity issues during translation.

[TPS_MANI_03612]{DRAFT} **Sufficient ASIL level of translation software** ⌈If the `SignalServiceTranslationEventProps.safeTranslation` equals true then the implementation of the translation software shall fulfill a sufficient ASIL.⌋*(RS_MANI_00063)*

[constr_3554]{DRAFT} **E2E protection configuration check** ⌈If the `SignalServiceTranslationEventProps.safeTranslation` equals true then the signal-based payload shall have an EndToEnd profile defined.⌋*()*

The current EndToEnd profiles for Classic platform rely on a periodic communication paradigm. For the translation software of the Adaptive platform this requires to know the specified period the payload has to be updated / checked for.

[TPS_MANI_03598]{DRAFT} **Expected check period of E2E-Protected payload** ⌈If the `RPortPrototype` has a `ReceiverComSpec.receptionProps.dataUpdatePeriod` defined for an `event` then the Adaptive Application calling the `ara::com` APIs shall check periodically for updates using the specified period.⌋*(RS_MANI_00063)*

[TPS_MANI_03599]{DRAFT} **Expected update period of E2E-Protected payload** ⌈If the `PPortPrototype` has a `SenderComSpec.transmissionProps.dataUpdatePeriod` defined for an `event` then the Adaptive Application calling the `ara::com` APIs shall periodically update the `event` using the specified period.⌋*(RS_MANI_00063)*

Class	**ReceptionComSpecProps**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	This meta-class defines a set of reception attributes which the application software is assumed to implement.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataUpdate Period	TimeValue	0..1	attr	This attribute defines the period in which the application shall check for updated data. This attribute is used for the configuration of the E2E protection, but may also indicate a general data reception period.
timeout	TimeValue	0..1	attr	This attribute defines the time interval after which the application shall assume that the to be received data reception has timed out, i.e. the respective data has not been received for that amount of time.

Table 11.11: ReceptionComSpecProps

Class	**TransmissionComSpecProps**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	This meta-class defines a set of transmission attributes which the application software is assumed to implement.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataUpdate Period	TimeValue	0..1	attr	This attribute defines the period in which the application is assumed to transmit the respective data.
minimumSend Interval	TimeValue	0..1	attr	This attribute defines the minimum interval between two consecutive transmissions of the respective data the application is assumed to ensure.
transmission Mode	TransmissionMode DefinitionEnum	0..1	attr	The attribute defines the mode in which the application is assumed to transmit the respective data.

Table 11.12: TransmissionComSpecProps

Enumeration	**TransmissionModeDefinitionEnum**
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication
Note	This meta-class defines possible settings for the transmission mode.
Literal	***Description***
cyclic	The data is assumed to be transmitted in a cyclic manner. The cycle is defined by dataUpdatePeriod. **Tags:**atp.EnumerationLiteralIndex=0
cyclicAndOn Change	The data is assumed to be transmitted in a cyclic manner (with cycle time dataUpdatePeriod) and additionally there may be arbitrary transmission if the data value changes (minimumSendInterval to be respected, if defined). **Tags:**atp.EnumerationLiteralIndex=2
triggered	The data is assumed to be transmitted in an arbitrary manner (minimumSendInterval to be respected, if defined). **Tags:**atp.EnumerationLiteralIndex=1

Table 11.13: TransmissionModeDefinitionEnum

11.9.1.1 Signal to Service

[TPS_MANI_03600]{DRAFT} **Signal-service-translation of E2E protected payload** ⌈Signal-service-translation shall check the end-to-end status of every received signal-based payload. If the E2E-Check returns *E_OK* for the received payload then the data shall be forwarded to the respective sending of the translation software.⌋*(RS_MANI_00063)*

Error handling:

[TPS_MANI_03601]{DRAFT} **Signal-service-translation of E2E protected payload - timeout handling** ⌈If no message is received within the specified message cycle time (timeout is detected), then no data shall be transmitted to the service-based part.⌋*(RS_MANI_00063)*

[TPS_MANI_03602]{DRAFT} **Signal-service-translation of E2E protected payload - error handling** ⌈If the E2E-Check returns any E2E error (not *E_OK*), then the service-based message shall reflect that E2E error.⌋*(RS_MANI_00063)*

Note: This is necessary to provide E2E information to the adaptive application and support an End-to-End view on the data exchange from sender to receiver / provider to consumer.

11.9.1.2 Service to Signal

[TPS_MANI_03603]{DRAFT} **Service-signal-translation of E2E protected payload** ⌈Signal-service-translation shall check the end-to-end status of every received service-oriented payload. If the E2E-Check returns *E_OK* for the received payload then the data can be forwarded to the respective sending of the translation software.⌋*(RS_MANI_00063)*

Error handling:

[TPS_MANI_03604]{DRAFT} **Service-signal-translation of E2E protected payload - timeout handling** ⌈If no message is received within the specified message cycle time (timeout is detected), then no data shall be transmitted to the signal-based part.⌋*(RS_MANI_00063)*

[TPS_MANI_03605]{DRAFT} **Service-signal-translation of E2E protected payload - error handling** ⌈If the service-oriented payload is handed over with any E2E error (not *E_OK*), then the newly created signal-based E2E protected message shall reflect that E2E error.⌋*(RS_MANI_00063)*

Note: This is necessary to provide E2E information to the receiving application and support an End-to-End view on the data exchange from sender to receiver/provider to consumer.

11.9.2 Security

In the context of Signal Service Translation the Secure Onboard Communication *SecOC* [31] is the major security technology. Further technologies (like *IPSec* or *TLS*) have not been included in the considerations for signal-service-translation.

The configuration of *SecOC* on the signal-based communication is defined by having the `ISignalTriggering` used in one of the signal-service-translation mappings refer to an `ISignal` and that `ISignal` is part of an `ISignalIPdu`. A `PduTriggering` of this `ISignalIPdu` is referenced by a `SecuredIPdu` in the role `payload`.

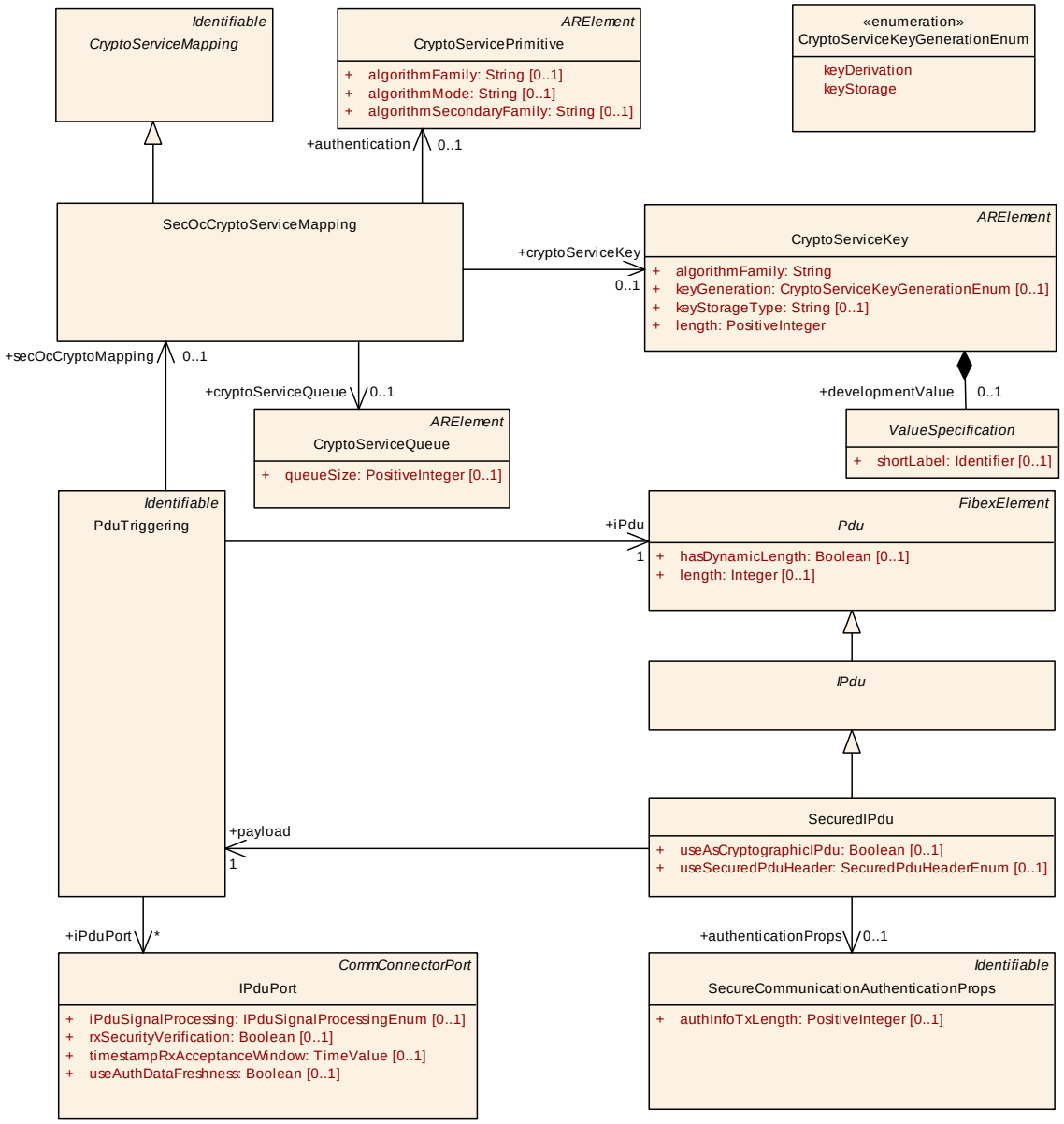


Figure 11.15: SecOC configuration

The `SecuredIPdu` defines all the details which are required to *generate* / *verify* the cryptographic information. The description of the configuration is provided in the System Template of Classic platform [17].

As the *SecOC* is highly embedded into the Classic platform architecture the signal-service-translation approach on security is to use the same architecture for its specification.

The APIs for *Freshness Management*, *Security Status Notification*, and *Security Status Control* are implementation specific for the time being[2].

[constr_3555]{DRAFT} **No support for `useAsCryptographicIPdu` is true** ⌈The signal-service-translation does not support the case where the `PduTriggering` is referencing a `SecuredIPdu` where the attribute `useAsCryptographicIPdu` is set to true.⌋*()*

The rationale for [constr_3555] is that the separate handling of two PDUs for the transport of one secured message is not supported by the SOME/IP protocol. In such cases the signal-service-translation has to be performed on a Classic platform gateway ECU.

[2]In an upcoming AUTOSAR release a detailed specification of *SecOC* in the Adaptive platform will be provided

12 Cross-FunctionalCluster interaction

[TPS_MANI_03268]{DRAFT} **Semantics of `FunctionalClusterInteractsWithFunctionalClusterMapping`** ⌈Abstract meta-class `FunctionalClusterInteractsWithFunctionalClusterMapping` provides an anchor for the specification of interaction between two functional clusters.

The identification of the interaction use case towards the functional cluster implementation is done by using an `InstanceSpecifier` of the concrete subclass of `FunctionalClusterInteractsWithFunctionalClusterMapping` that is used to define the concrete interaction.⌋*()*

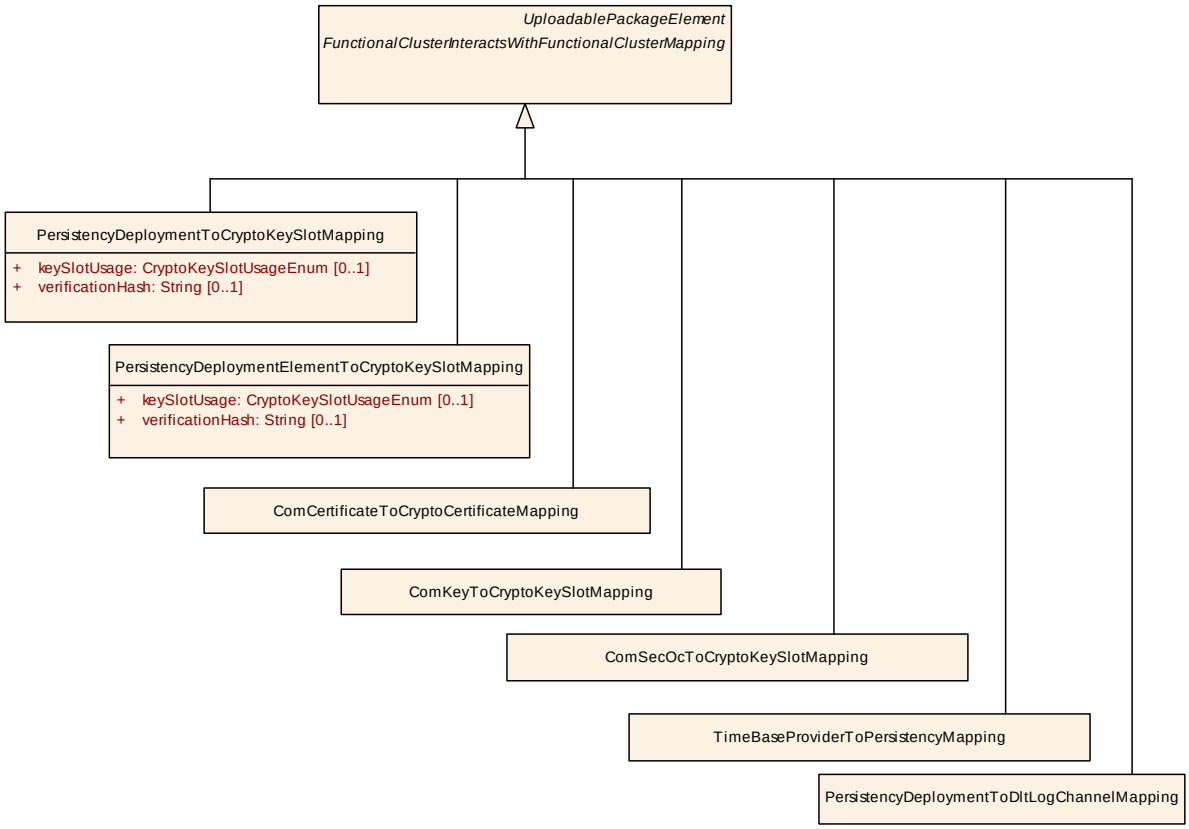


Figure 12.1: FunctionalClusterInteractsWithFunctionalClusterMapping Overview

Class	***FunctionalClusterInteractsWithFunctionalClusterMapping*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment
Note	This meta-class identifies a relation between functional clusters on the adaptive platform such one functional cluster can call APIs of the other functional cluster. **Tags:**atp.Status=draft
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*

▽

△

Class	***FunctionalClusterInteractsWithFunctionalClusterMapping*** (abstract)			
Subclasses	ComCertificateToCryptoCertificateMapping, ComKeyToCryptoKeySlotMapping, ComSecOcToCryptoKeySlotMapping, PersistencyDeploymentElementToCryptoKeySlotMapping, PersistencyDeploymentToCryptoKeySlotMapping, PersistencyDeploymentToDltLogChannelMapping, TimeBaseProviderToPersistencyMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 12.1: FunctionalClusterInteractsWithFunctionalClusterMapping

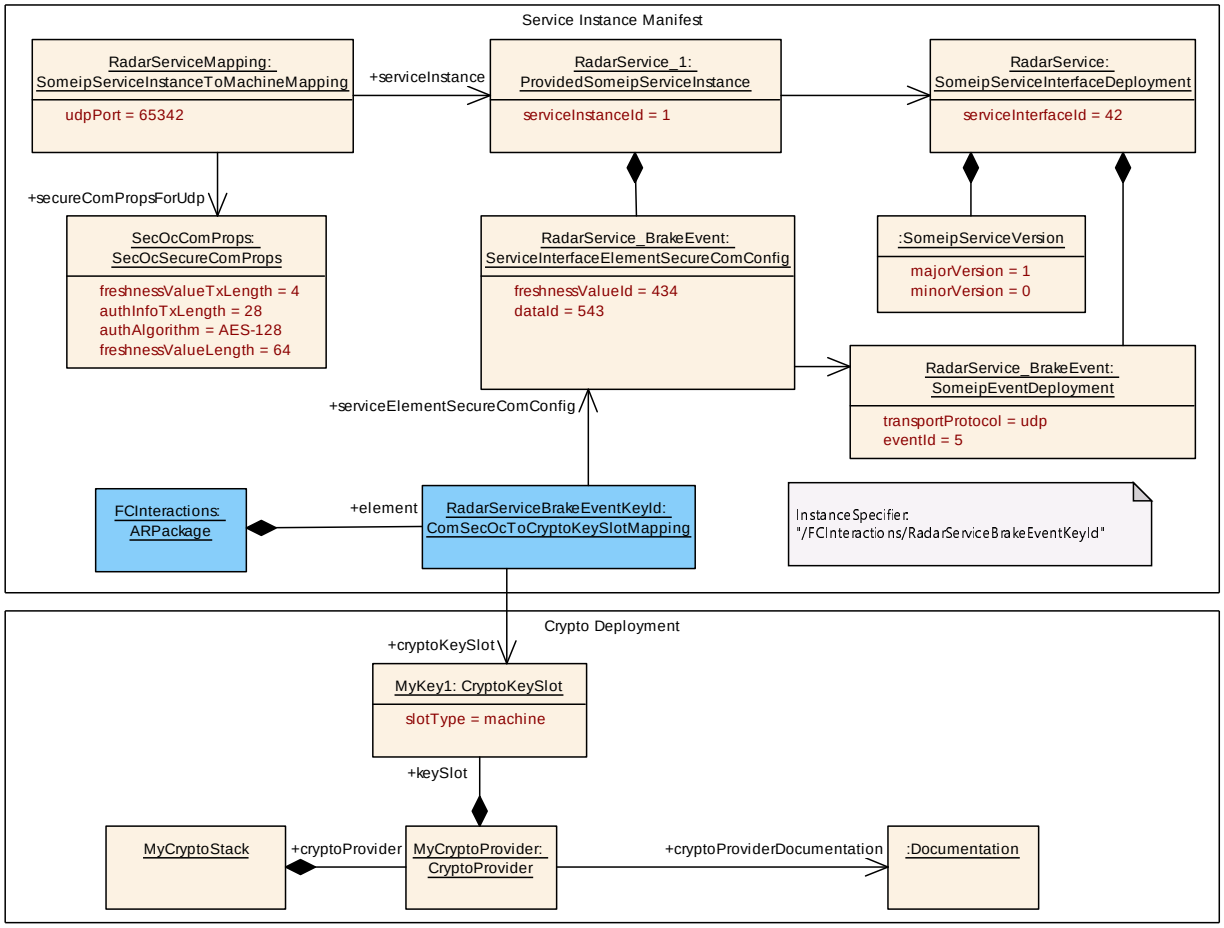


Figure 12.2: Example for the usage of `FunctionalClusterInteractsWithFunctionalClusterMapping`

In case of an application the model-path to an `PortPrototype` that is referencing a specific `PortInterface` defines the `InstanceSpecifier` that is used as identification towards the functional cluster implementation.

In case of a FunctionalCluster interaction the `PortPrototype` is not available. Therefore the path to a mapping element that is derived from `FunctionalClusterInteractsWithFunctionalClusterMapping` is used to define the `InstanceSpecifier` that is used in the API call.

The following figure shows the supported FunctionalClusterInteractsWith-FunctionalClusterMapping subclasses that are available in the model.

The following Figure 12.2 shows an example for the usage of FunctionalCluster-InteractsWithFunctionalClusterMapping. In this example, the Service Instance Manifest describes a configuration for a ProvidedSomeipServiceInstance that contains an Event that is protected by SecOC.

Inside of the Service Instance Manifest the "RadarServiceBrakeEventKeyId" represents the FunctionalClusterInteractsWithFunctionalClusterMapping that is located in the ARPackage "FCInteractions".

These two elements in the manifest define the InstanceSpecifier "\FCInteractions\RadarServiceBrakeEventKeyId" that is used in the Crypto API call from the Communication Management. This InstanceSpecifier is resolved to a concrete CryptoKeySlot with the information that is available in the Crypto Deployment.

12.1 ComCertificateToCryptoCertificateMapping

[TPS_MANI_03269]{DRAFT} **Semantics of ComCertificateToCryptoCertificateMapping** ⌈The meta-class ComCertificateToCryptoCertificateMapping provides an anchor for the specification of interaction between the COM FunctionalCluster and the Crypto FunctionalCluster and is used to map a CryptoServiceCertificate defined in COM to a CryptoCertificate defined in the Crypto Stack.⌋*()*

Class	**ComCertificateToCryptoCertificateMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class maps the CryptoServiceCertificate defined in the COM deployment to the Crypto Certificate defined in the Crypto Stack. **Tags:** atp.Status=draft atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
crypto Certificate	CryptoCertificate	0..1	ref	Reference to the CryptoCertifcate in the Crypto Stack. **Tags:**atp.Status=draft
cryptoService Certificate	CryptoService Certificate	0..1	ref	Reference to the cryptoServiceCertificate in the Com deployment **Tags:**atp.Status=draft

Table 12.2: ComCertificateToCryptoCertificateMapping

Figure 12.3 shows the ComCertificateToCryptoCertificateMapping used to assign a CryptoServiceCertificate defined in the TLS configuration to a CryptoCertificate in the Crypto Stack.

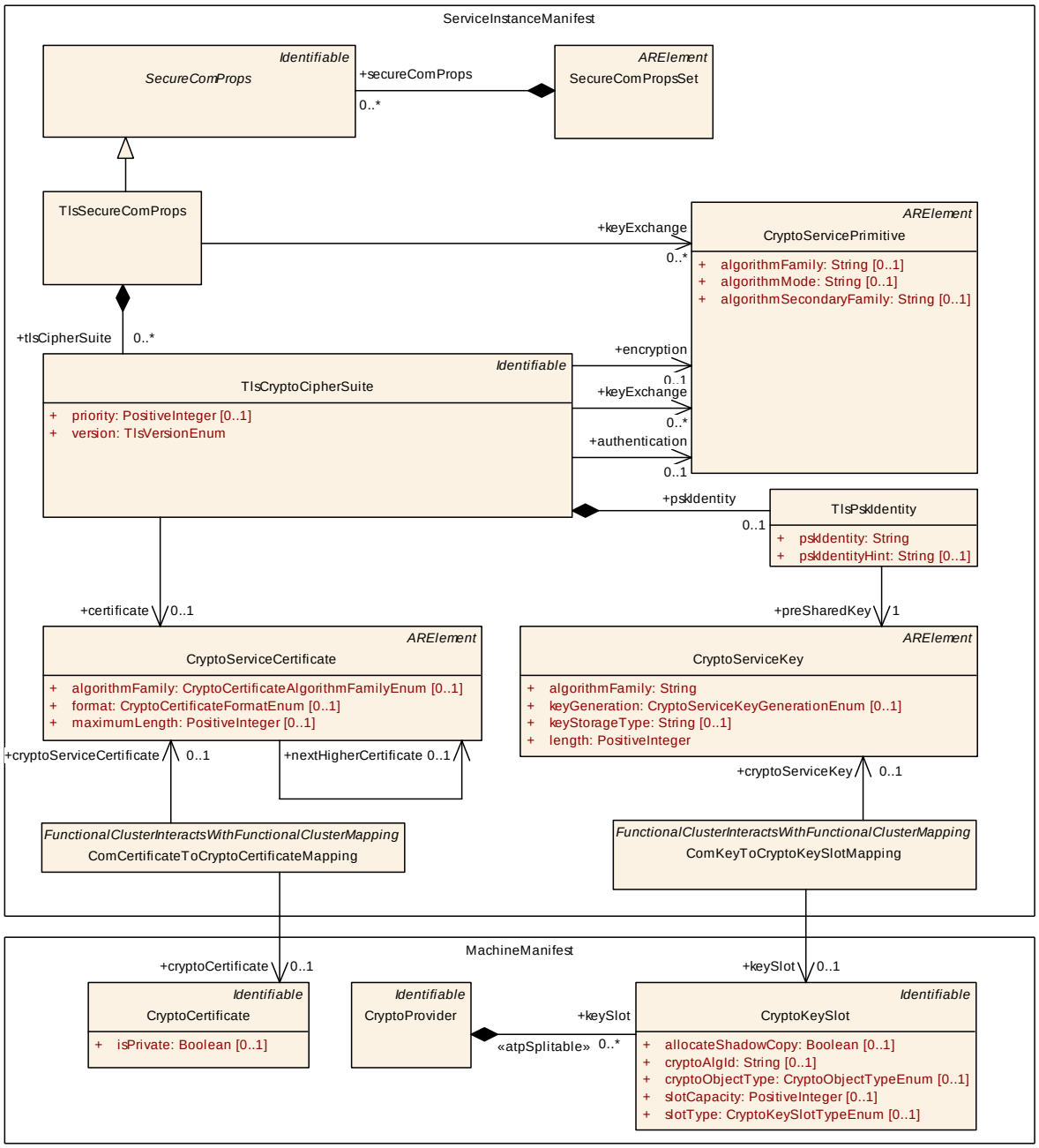


Figure 12.3: Mapping of TLS config elements to crypto objects in the crypto stack

12.2 ComKeyToCryptoKeySlotMapping

[TPS_MANI_03270]{DRAFT} **Semantics of `ComKeyToCryptoKeySlotMapping`**
⌈The meta-class `ComKeyToCryptoKeySlotMapping` provides an anchor for the specification of interaction between the COM FunctionalCluster and the Crypto FunctionalCluster and is used to map a `CryptoServiceKey` defined in COM to a `CryptoKeySlot` defined in the Crypto Stack.⌋()

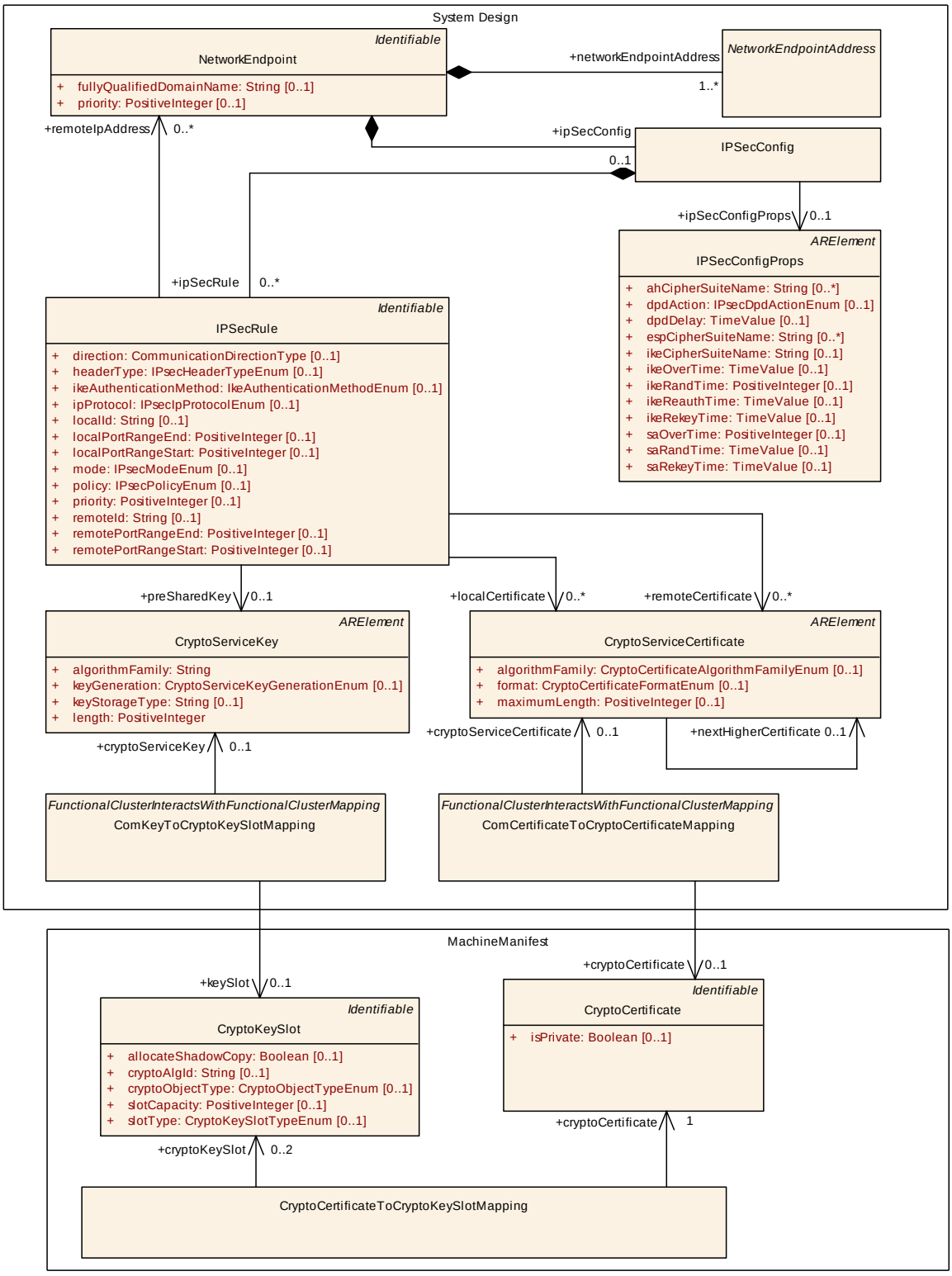


Figure 12.4: Mapping of IPsec config elements to crypto objects in the crypto stack

Class	ComKeyToCryptoKeySlotMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class maps the CryptoServiceKey defined in the COM deployment to the CryptoKeySlot defined in the Crypto Stack. **Tags:** atp.Status=draft atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
cryptoServiceKey	CryptoServiceKey	0..1	ref	Reference to the cryptoServiceKey in the Com deployment **Tags:**atp.Status=draft
keySlot	CryptoKeySlot	0..1	ref	Reference to the CryptoKeySlot in the Crypto Stack. **Tags:**atp.Status=draft

Table 12.3: ComKeyToCryptoKeySlotMapping

Figure 12.4 shows the `ComKeyToCryptoKeySlotMapping` used to assign a `CryptoServiceKey` defined in the IPsec configuration to `CryptoKeySlot` in the Crypto Stack.

12.3 ComSecOcToCryptoKeySlotMapping

[TPS_MANI_03271]{DRAFT} **Semantics of `ComSecOcToCryptoKeySlotMapping`** ⌈The meta-class `ComSecOcToCryptoKeySlotMapping` provides an anchor for the specification of interaction between the COM FunctionalCluster and the Crypto FunctionalCluster and is used to map a `ServiceInterfaceElementSecureComConfig` defined in COM to a `CryptoKeySlot` defined in the Crypto Stack.⌋*()*

Class	ComSecOcToCryptoKeySlotMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class maps the ServiceElementSecureComConfig defined in the COM deployment to the CryptoKeySlot defined in the Crypto Stack. **Tags:** atp.Status=draft atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
cryptoKeySlot	CryptoKeySlot	0..1	ref	Reference to the CryptoKeySlot in the Crypto Stack. **Tags:**atp.Status=draft
serviceElementSecureComConfig	ServiceInterfaceElementSecureComConfig	0..1	ref	Reference to the ServiceInterfaceElementSecureComConfig element in the COM config. **Tags:**atp.Status=draft

Table 12.4: ComSecOcToCryptoKeySlotMapping

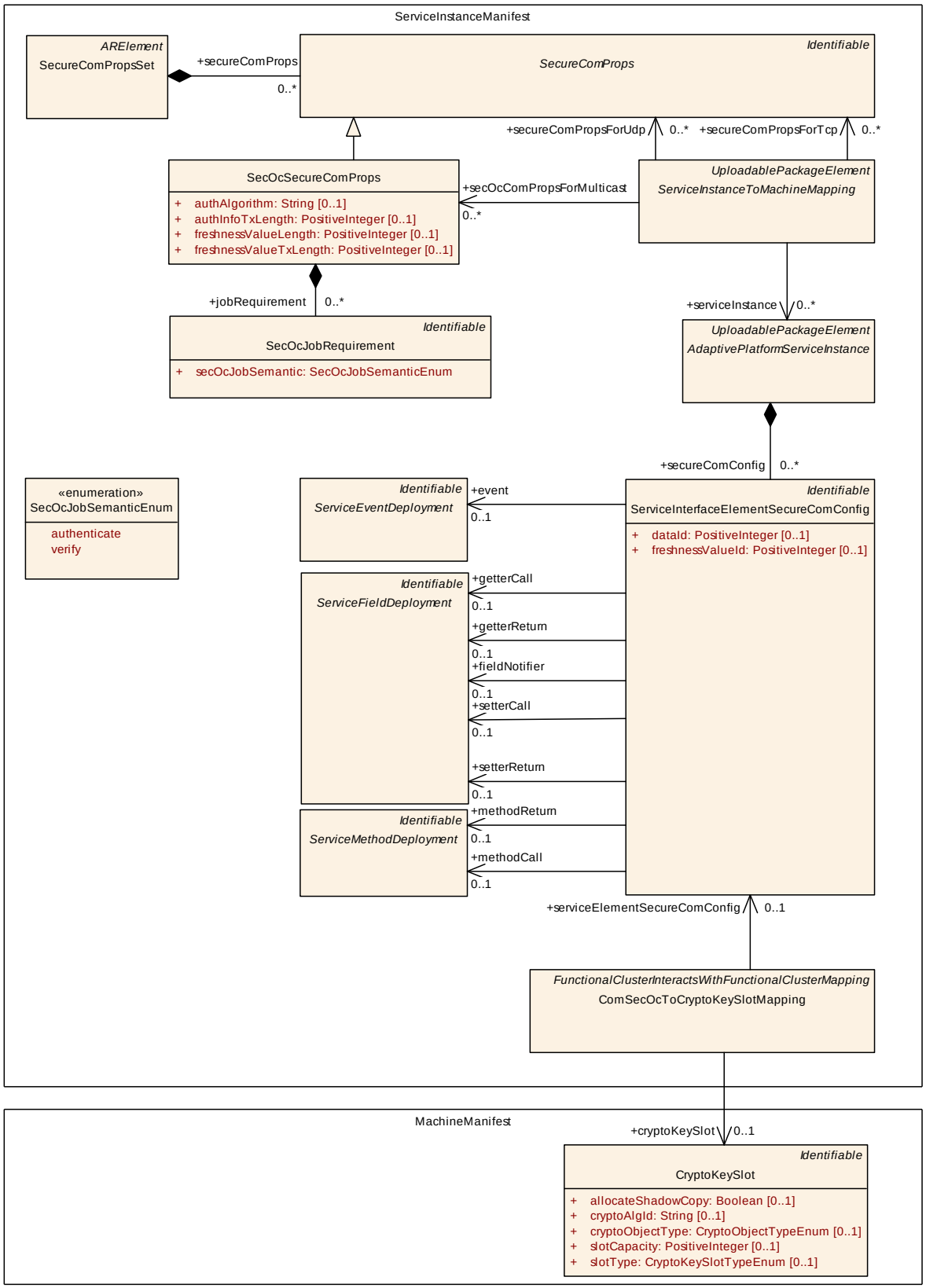


Figure 12.5: Mapping of SecOC config elements to crypto objects in the crypto stack

Figure 12.5 shows the ComSecOcToCryptoKeySlotMapping used to assign a ServiceInterfaceElementSecureComConfig defined in the SecOC configuration to CryptoKeySlot in the Crypto Stack.

12.4 PersistencyDeploymentToCryptoKeySlotMapping

[TPS_MANI_03272]{DRAFT} **Semantics of PersistencyDeploymentToCryptoKeySlotMapping** ⌈The meta-class PersistencyDeploymentToCryptoKeySlotMapping provides an anchor for the specification of interaction between the Persistency FunctionalCluster and the Crypto FunctionalCluster and is used to map a PersistencyDeployment defined in Persistency to a CryptoKeySlot defined in the Crypto Stack.⌋*()*

Class	**PersistencyDeploymentToCryptoKeySlotMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to define a mapping between the PersistencyDeployment and a CryptoKeySlot. **Tags:** atp.Status=draft atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cryptoKeySlot	CryptoKeySlot	0..1	ref	This reference represents the mapped CryptoKeySlot. **Tags:**atp.Status=draft
keySlotUsage	CryptoKeySlotUsage Enum	0..1	attr	This attribute defines the role of the keySlot assignment.
persistency Deployment	PersistencyDeployment	1	ref	This reference represents the mapped Persistency Deployment. **Tags:**atp.Status=draft
verificationHash	String	0..1	attr	This attribute defines the hash of the storage used in case of verification.

Table 12.5: PersistencyDeploymentToCryptoKeySlotMapping

Enumeration	**CryptoKeySlotUsageEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment
Note	This enum defines the possible roles of the keySlotUsage. **Tags:**atp.Status=draft
Literal	***Description***
encryption	Key slot usage for enryption **Tags:**atp.EnumerationLiteralIndex=1
verification	Key slot usage for verification **Tags:**atp.EnumerationLiteralIndex=0

Table 12.6: CryptoKeySlotUsageEnum

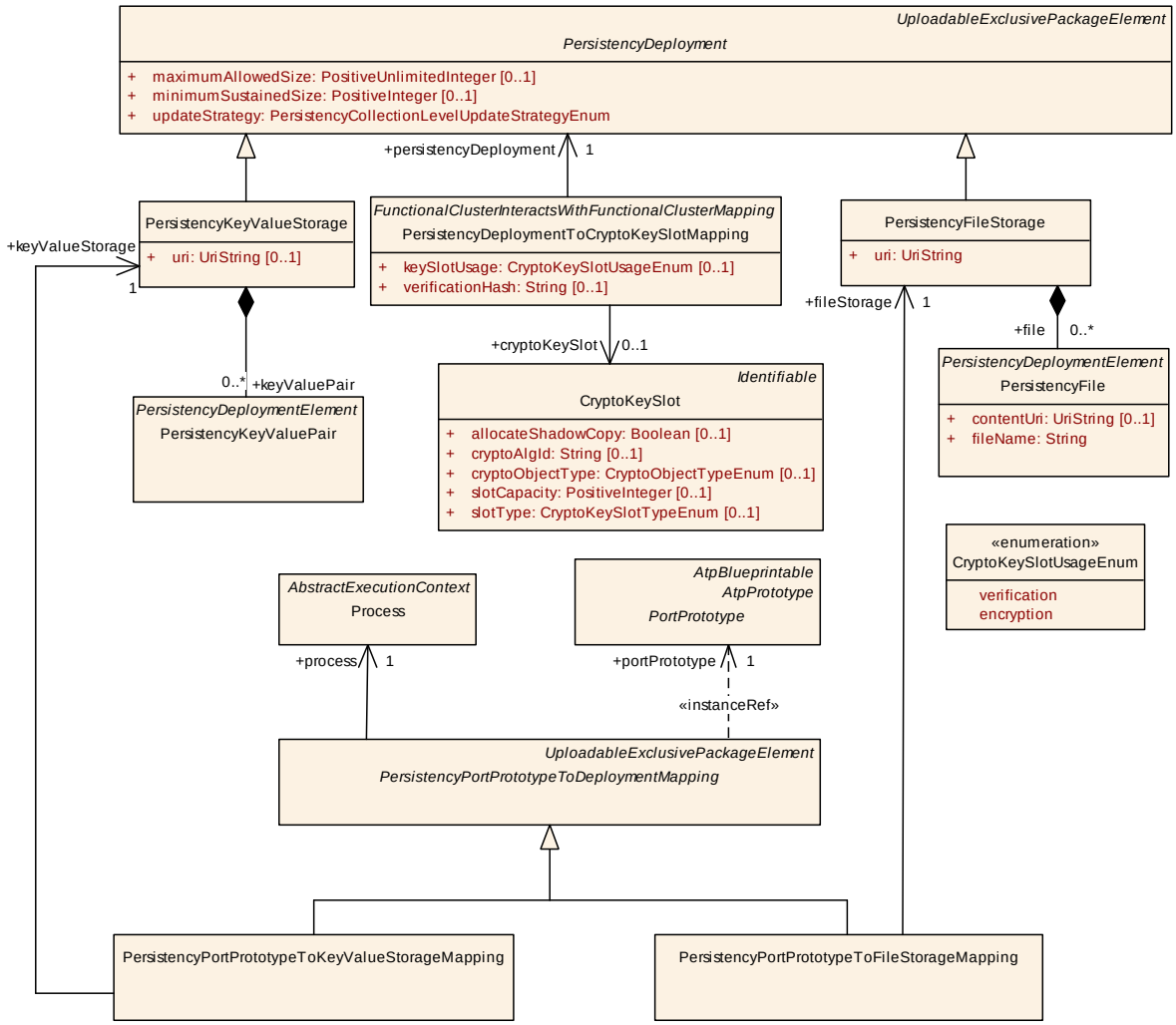


Figure 12.6: Mapping of PersistencyDeployment to crypto objects in the crypto stack

12.5 PersistencyDeploymentElementToCryptoKeySlotMapping

[TPS_MANI_03273]{DRAFT} **Semantics of `PersistencyDeploymentElementToCryptoKeySlotMapping`** ⌈The meta-class `PersistencyDeploymentElementToCryptoKeySlotMapping` provides an anchor for the specification of interaction between the Persistency FunctionalCluster and the Crypto FunctionalCluster and is used to map a `PersistencyDeploymentElement` defined in Persistency to a `CryptoKeySlot` defined in the Crypto Stack.⌋*()*

Class	**PersistencyDeploymentElementToCryptoKeySlotMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to define a mapping between the PersistencyDeploymentElement and a CryptoKeySlot. **Tags:** atp.Status=draft atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cryptoKeySlot	CryptoKeySlot	0..1	ref	This reference represents the mapped CryptoKeySlot. **Tags:**atp.Status=draft
keySlotUsage	CryptoKeySlotUsage Enum	0..1	attr	This attribute defines the role of the keySlot assignment.
persistency Deployment Element	PersistencyDeployment Element	0..1	ref	This reference represents the mapped Persistency Deployment. **Tags:**atp.Status=draft
verificationHash	String	0..1	attr	This attribute defines the hash of the storage used in case of verification.

Table 12.7: PersistencyDeploymentElementToCryptoKeySlotMapping

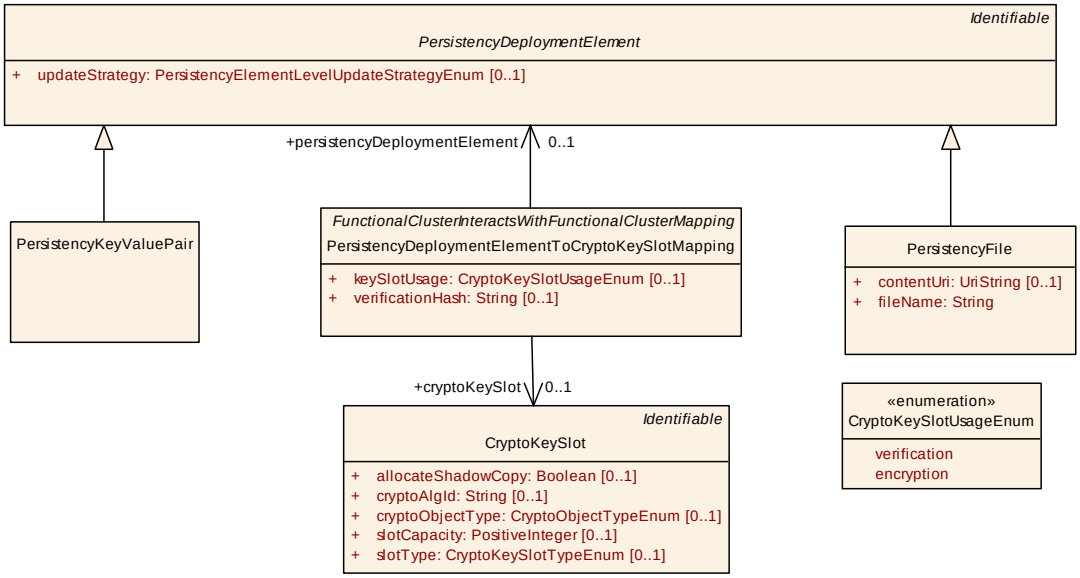


Figure 12.7: Mapping of PersistencyDeploymentElement to crypto objects in the crypto stack

12.6 PersistencyDeploymentToDltLogChannelMapping

[TPS_MANI_03276]{DRAFT} **Semantics of `PersistencyDeploymentToDltLogChannelMapping`** ⌈The meta-class `PersistencyDeploymentToDltLogChannelMapping` provides an anchor for the specification of interaction between the Persistency FunctionalCluster and Log & Trace and is used to map a

PersistencyDeployment defined in Persistency to a DltLogChannel defined in the LogAndTraceInstantiation.⌋()

Class	PersistencyDeploymentToDltLogChannelMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::LogAndTrace			
Note	This meta-class represents the ability to define a mapping between the PersistencyDeployment and a Dlt LogChannel. **Tags:** atp.Status=draft atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dltLogChannel	DltLogChannel	0..1	ref	This reference represents the mapped DltLogChannel. **Tags:**atp.Status=draft
persistency Deployment	PersistencyDeployment	0..1	ref	This reference represents the mapped Persistency Deployment. **Tags:**atp.Status=draft

Table 12.8: PersistencyDeploymentToDltLogChannelMapping

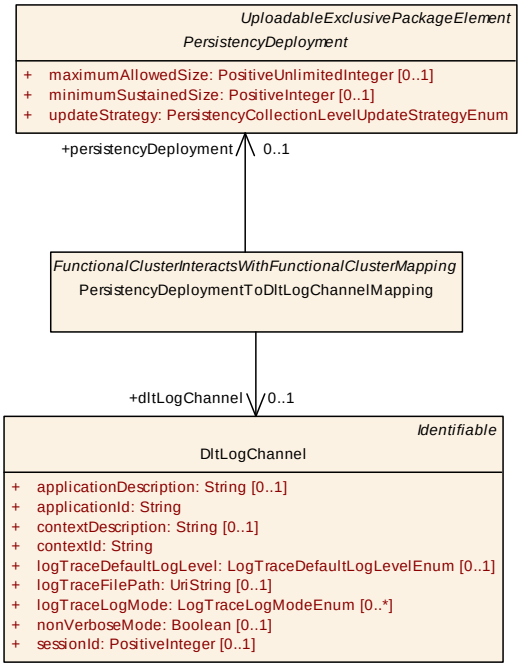


Figure 12.8: Mapping of PersistencyDeployment to DltLogChannel

13 REST

13.1 REST Design

13.1.1 Overview

Important note: the AUTOSAR SWS REST [5] defines a low-level API for REST-based communication. The content of this chapter, on the other hand, applies for the configuration of a not-yet standardized API on top of the ara::rest API.

In line with the target application domains of the *AUTOSAR adaptive platform* it can be expected that software will have use case to interact with generic web services inside and outside the vehicle.

Obviously, the communication partners need to agree on the applied communication approach to make this happen.

In other words, while it would be technically feasible to implement web services based on the existence of ServiceInterfaces it is still not very likely to happen for services that are completely outside the typical automotive domain and which have no incentive to embrace the communication approach of the *AUTOSAR adaptive platform*.

Therefore, the only viable option seems to extend the communications capabilities of the adaptive AUTOSAR stack to talk to web services in their “native language”.

The conclusion to adopt web service communication approach does not only extend to the actual communication and transport conventions but also affects the way how information conveyed between a vehicle and a web service is described.

In order to fully implement a communication paradigm for information exchange with web services, the *AUTOSAR adaptive platform* needs to adopt conventions of data description that are typically supported by web services.

As a matter of fact, web services don’t dive into data semantics nearly as deep as this is done in a typical automotive software and therefore seamlessly supported by the AUTOSAR meta-model. Consequently, AUTOSAR needs to define an alternative approach to data definition that matches with the conventions established for web services.

Consequently, the approach to define ApplicationDataTypes and their ImplementationDataType counterparts is not applicable for this case.

But still, the general AUTOSAR approach to structure application software into the definition of ApplicationSwComponentTypes that interact with the outside world via the existence of aggregated PortPrototypes applies also for software that interacts with web services.

In other words, interaction with web services need to be placed on the definition of a specific subclass of PortInterface in order to conform to the above mentioned statement.

The concrete definition of such a subclass of PortInterface requires a more specific understanding of the typical interaction patterns of web services.

While it is safe to conclude that the web breeds new technologies on nearly a weekly basis, there is still some stable core on which the modeling in AUTOSAR can rely on.

This stable core onto which the AUTOSAR modeling approach shall be based has been identified as the so-called "**Representational State Transfer**" [32] (a.k.a. REST) pattern.

Fundamentally, the REST approach requires a stateless communication among server and client, i.e. only data can be communicated.

The call of a method or operation (which is otherwise supported by means of the ServiceInterface or ClientServerInterface) is expressly out of scope.

[TPS_MANI_01103]{DRAFT} **Three-level approach to REST modeling** ⌈The conversion of the REST pattern, as far as modeling is concerned, into AUTOSAR assumes a three-level structure:

Service This level represents the definition of an entire REST service.

In the AUTOSAR meta-model, this level is represented by meta-class RestServiceInterface.

Resource This level represents a resource in the context of the service. A resource can be used to structure the content of a service according to a given conceptual understanding of the semantics of the service.

For example, if a *sound mixer* were a service then it could make sense to define *audio source*, *output device*, etc as resources of the service. There can still be several sources and several output devices.

In the AUTOSAR meta-model, the resource level is represented by meta-class RestResourceDef.

Element The final level represents the definition of actual data with properties in the context of a resource. In the context of the above mentioned example of a *sound mixer* the element level of the *output device* resource could be populated by *volume*, *volume step-size*, *status*, etc.

In the AUTOSAR meta-model, the element level is represented by meta-class RestElementDef.

⌋*(RS_MANI_00033)*

The three-level approach described in [TPS_MANI_01103] is depicted in Figure 13.1.

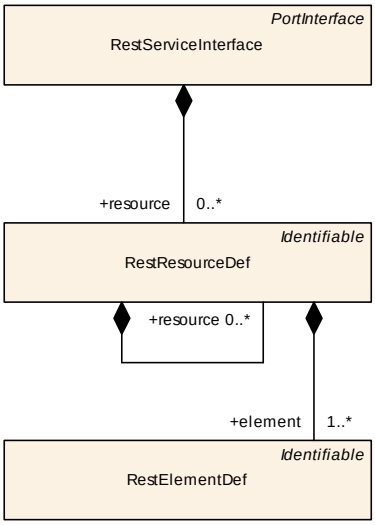


Figure 13.1: Big picture to REST modeling

Rest services are identified by means of a URI. The details of how the URI is created for a specific REST service can (because of the possibility of multiple instantiation of SwComponentTypes that aggregate PortPrototypes typed by a RestServiceInterface) only be resolved in the deployment phase where the specific instances are known.

The details of what makes a URI for a REST service as well as a description of how elements of the URI are sourced can be found in section 13.2.

Please note that in the domain of web services a service description is often provided in JSON format. The description of REST services in this chapter introduces the description of REST services to AUTOSAR and this has the consequence that ARXML has to be used for this purpose.

However, AUTOSAR does not oblige the usage of ARXML on the target platform, it only says that there shall be a point in time when the final model has to be available as ARXML and that exchange of AUTOSAR models shall only be done in ARXML format.

From the point of finalization going forward, proprietary conversions into whatever format for the sole purpose of uploading to a target platform is permitted.

Conversely, it is totally conceivable to create a conversion tool that takes an existing service description in JSON format and converts it into the ARXML representation described in this chapter.

Please note further that REST typically supports a filtering of information on the server, i.e. the client can apply a filter to only obtain the part of information on the server that passes the filter.

This filtering approach fully happens at run-time, there is no need to configure anything in the model in order to support the filtering of information on the server.

13.1.2 REST Service Interface

As depicted in Figure 13.2, RestServiceInterface is derived from PortInterface and can therefore be taken to type a PortPrototype.

In other words, the definition of a REST service creates a binding contract for the implementation of the ApplicationSwComponentType that aggregates a PortPrototype typed by a RestServiceInterface.

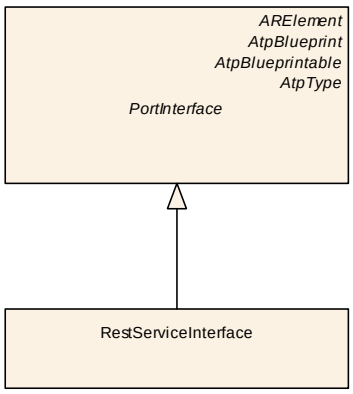


Figure 13.2: Modeling of the REST service

[TPS_MANI_01105]{DRAFT} **Semantics of RestServiceInterface** ⌈A PortPrototype used to interact by means of the REST pattern with a web service shall be typed by RestServiceInterface.⌋ *(RS_MANI_00033)*

Class	**RestServiceInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents a REST service. **Tags:** atp.Status=draft atp.recommendedPackage=RestServiceInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
resource	RestResourceDef	*	aggr	This aggregation represents the collectioin of resources owned by the enclosing REST service. **Tags:**atp.Status=draft

Table 13.1: RestServiceInterface

13.1.3 REST Resource

[TPS_MANI_01120]{DRAFT} **Recursive definition of RestResourceDef** ⌈The definition of RestResourceDef supports the aggregation of other RestResourceDef. In other words, it is possible to create a nested definition of RestResourceDefs.⌋ *(RS_MANI_00033)*

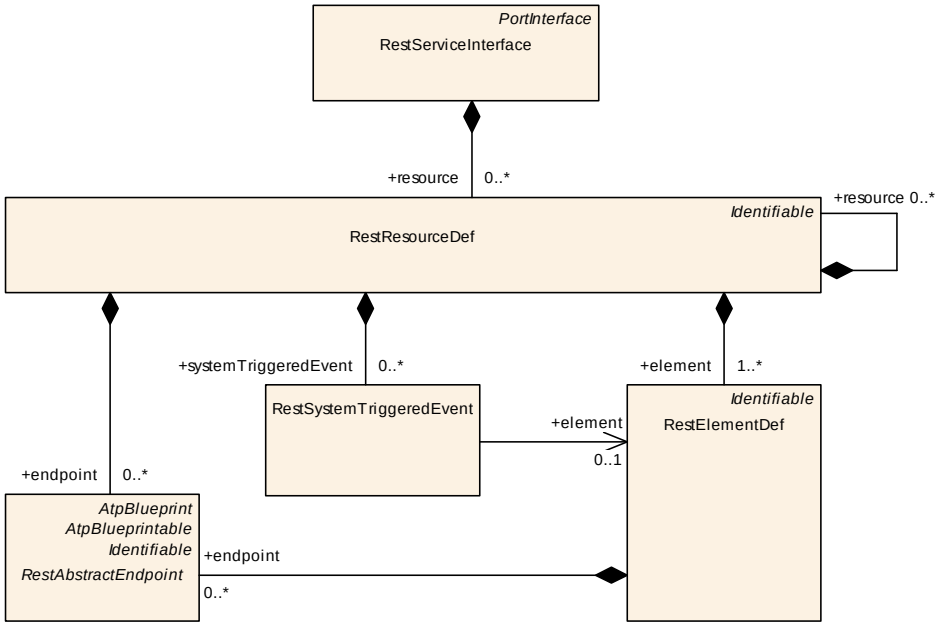


Figure 13.3: Modeling of the REST resource level

Class	**RestResourceDef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents a resource inside a REST service. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
element	RestElementDef	1..*	aggr	This aggregation represents the elements of a resource. **Tags:**atp.Status=draft
endpoint	RestAbstractEndpoint	*	aggr	This aggregation represents the collection of endpoints on the resource level. **Tags:**atp.Status=draft
resource	RestResourceDef	*	aggr	This aggregation represenst the ability to create nested resource levels. **Tags:**atp.Status=draft
system TriggeredEvent	RestSystemTriggered Event	*	aggr	This represents the collection of system triggered events for the enclosing resource. **Tags:**atp.Status=draft

Table 13.2: RestResourceDef

[TPS_MANI_01121]{DRAFT} **Semantics of `RestResourceDef.endpoint`** ⌈It is possible to define the `API` that shall be available for a specific `RestResourceDef`. For this purpose the aggregation of `RestAbstractEndpoint` in the role `endpoint` shall be used.

In particular the following concrete API elements (that directly correspond to the eponymous HTTP verbs) can be modeled:

GET For this purpose meta-class `RestEndpointGet` shall be used.

PUT For this purpose meta-class `RestEndpointPut` shall be used.

POST For this purpose meta-class RestEndpointPost shall be used.

DELETE For this purpose meta-class RestEndpointDelete shall be used.

⌋(*RS_MANI_00033*)

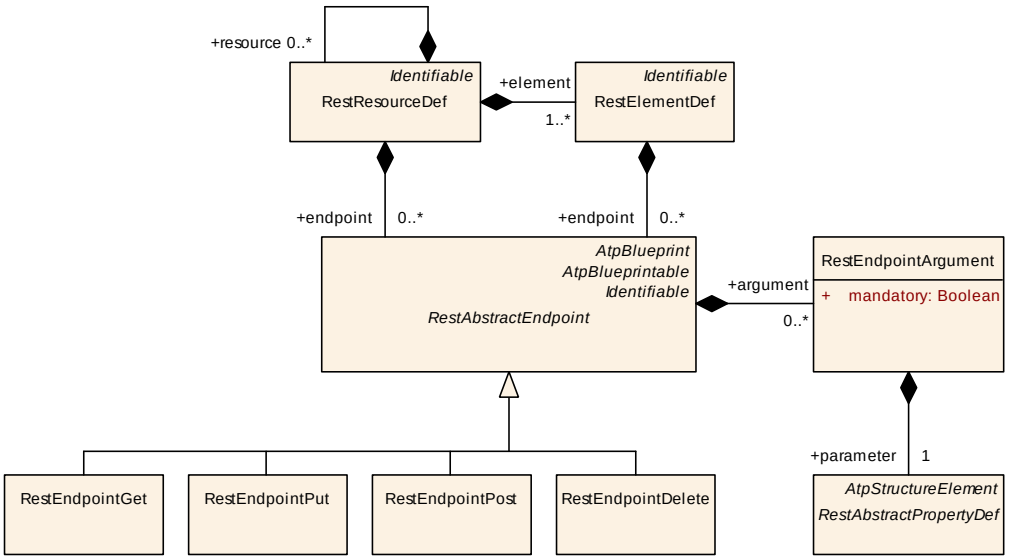


Figure 13.4: Modeling of the REST endpoints

[TPS_MANI_01122]{DRAFT} **Arguments to endpoints** ⌈In many cases a concrete subclass of RestAbstractEndpoint needs arguments to fulfill its intended semantics. An argument to such an endpoint can be defined by means of the aggregation of RestEndpointArgument in the role RestAbstractEndpoint.argument. Arguments can be required to exist or may be optional. This question is clarified by means of attribute RestEndpointArgument.mandatory.

The actual "payload" of the argument is not defined by RestEndpointArgument itself, for this the aggregation RestEndpointArgument.parameter shall be used.⌋ (*RS_MANI_00033*)

Class	***RestAbstractEndpoint*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class acts as a base class for the definition of endpoints within REST services. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	RestEndpointDelete, RestEndpointGet, RestEndpointPost, RestEndpointPut			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
argument	RestEndpointArgument	*	aggr	Some endpoints can require a list of arguments. **Tags:**atp.Status=draft

Table 13.3: RestAbstractEndpoint

Class	**RestEndpointPut**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to model a REST endpoint with PUT semantics. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *RestAbstractEndpoint*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 13.4: RestEndpointPut

Class	**RestEndpointGet**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to model a REST endpoint with GET semantics. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *RestAbstractEndpoint*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 13.5: RestEndpointGet

Class	**RestEndpointPost**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to model a REST endpoint with POST semantics. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *RestAbstractEndpoint*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 13.6: RestEndpointPost

Class	**RestEndpointDelete**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to model a REST endpoint with DELETE semantics. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *RestAbstractEndpoint*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 13.7: RestEndpointDelete

Class	**RestEndpointArgument**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to define an argument for a REST endpoint. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
mandatory	Boolean	1	attr	This attribute defines whether the argument is mandatory or whether it could be left out. **Tags:**atp.Status=draft
parameter	RestAbstractProperty Def	1	aggr	This aggregation represents the concrete kind of argument to be used. **Tags:**atp.Status=draft

Table 13.8: RestEndpointArgument

[TPS_MANI_01123]{DRAFT} **System Triggered Event** ⌈A `RestSystemTriggeredEvent` aggregated in the role `RestResourceDef.systemTriggeredEvent` can be modeled to indicate that a notifier for changes of the specific `RestElementDef` referenced in the role `RestSystemTriggeredEvent.element` shall be created.

By this means the server is able to inform any respectively configured client about changes of the referenced element.⌋ *(RS_MANI_00033)*

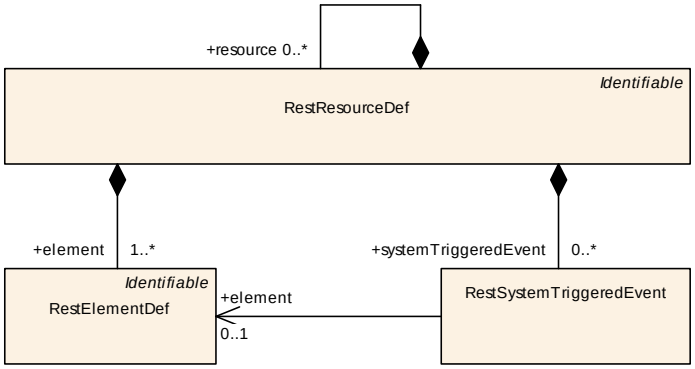


Figure 13.5: Modeling of the REST system triggered event

Class	**RestSystemTriggeredEvent**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to identify an element such that at runtime an event is generated when the value of the reference element changes. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
element	RestElementDef	0..1	ref	This reference represent the element that is linked to the system triggered event. **Tags:**atp.Status=draft

Table 13.9: RestSystemTriggeredEvent

13.1.4 REST Element

[TPS_MANI_01124]{DRAFT} **Semantics of `RestElementDef`** ⌈Meta-class `RestElementDef` represents the definition of data within a REST service. The specific definition of the data is done by way of aggregating so-called properties, i.e. `RestElementDef` aggregates `RestAbstractPropertyDef` in the role `property`.⌋ *(RS_MANI_00033)*

Class	**RestElementDef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents an element of a resource that in turn is owned by a REST service. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
endpoint	RestAbstractEndpoint	*	aggr	This aggregation represents the definition of endpoints on the object level. **Tags:**atp.Status=draft
property	RestAbstractProperty Def	1..*	aggr	This aggregation represents the collection of non-obligatory properties of the element level in a REST service. **Tags:**atp.Status=draft

Table 13.10: RestElementDef

Class	***RestAbstractPropertyDef*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class acts as an abstract subclass for the definition of properties owned by the element level of a REST service definition. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	RestArrayPropertyDef, *RestPrimitivePropertyDef*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 13.11: RestAbstractPropertyDef

As depicted by Figure 13.6, there is a certain variety of ways in which the properties of a REST element can be described.

However, the expressiveness of this description is in no way comparable to the richness of the semantics of an `ApplicationDataType` or a `CppImplementationDataType`.

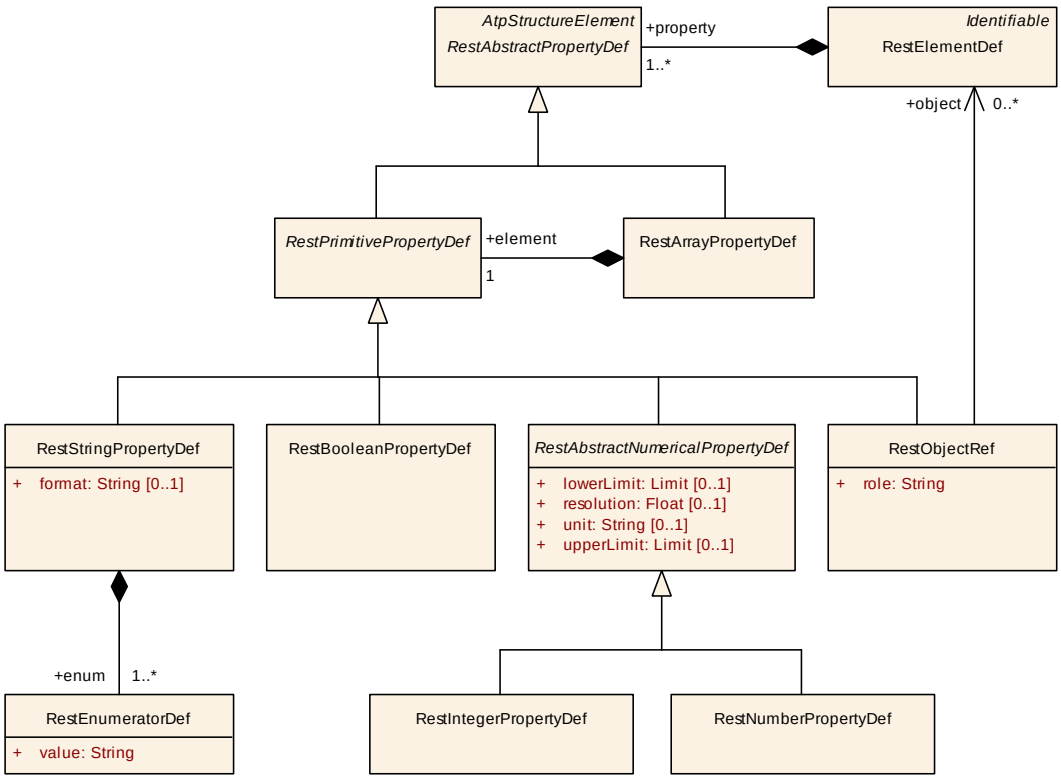


Figure 13.6: Modeling of the REST elements

[TPS_MANI_01125]{DRAFT} **Properties of REST elements can either be primitive or have array semantics** ⌈The properties of REST elements can either be primitive or have array semantics.

There is no support for the creation of structures nor is the nesting of property definitions with array semantics supported.

This aspect is already clarified by the model (`RestArrayPropertyDef` directly aggregates `RestPrimitivePropertyDef`) and does not need to be expressed by a written constraint.⌋ *(RS_MANI_00033)*

Class	***RestPrimitivePropertyDef*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class acts as an abstract base class for the definition of primitive properties of elements of a REST service. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *RestAbstractPropertyDef*			
Subclasses	*RestAbstractNumericalPropertyDef*, RestBooleanPropertyDef, RestObjectRef, RestStringPropertyDef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 13.12: RestPrimitivePropertyDef

Class	RestArrayPropertyDef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to define a property of an element of a rest service where the property is supposed to represent an array of other primitive properties. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *RestAbstractPropertyDef*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
element	RestPrimitiveProperty Def	1	aggr	This aggregation represents the definition of the base element type of the array property **Tags:**atp.Status=draft

Table 13.13: RestArrayPropertyDef

Class	RestBooleanPropertyDef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to define a REST property with boolean semantics. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *RestAbstractPropertyDef*, *RestPrimitivePropertyDef*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 13.14: RestBooleanPropertyDef

[TPS_MANI_01126]{DRAFT} **Definition of string properties** ⌈Properties with string semantics can be defined by means of `RestStringPropertyDef`.

In many cases, the intention will be to only allow a certain number of values within the string property and define the potential values of the string property directly by the string property itself.

For this purpose, `RestStringPropertyDef` aggregates `RestEnumeratorDef` in the role `enum` that in turn allows for the definition of the predefined value by way of attribute `value`.⌋*(RS_MANI_00033)*

Class	RestStringPropertyDef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to define a REST property with string semantics. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *RestAbstractPropertyDef*, *RestPrimitivePropertyDef*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
enum	RestEnumeratorDef	1..*	aggr	This aggregation represents the collection of enumerators for the enclosing string property. **Tags:**atp.Status=draft

▽

△

Class	**RestStringPropertyDef**			
format	String	0..1	attr	This attribute can be used to define a specific format that the value of the string property shall be conform with. **Tags:**atp.Status=draft

Table 13.15: RestStringPropertyDef

Class	**RestEnumeratorDef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to define enumerator values that can be taken as a the value of the enclosing string property. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
value	String	1	attr	This attribute represents the ability to assign a value to an enumerator. **Tags:**atp.Status=draft

Table 13.16: RestEnumeratorDef

[TPS_MANI_01127]{DRAFT} **Limited support for data semantics in `RestAbstractNumericalPropertyDef`** ⌈Meta-class `RestAbstractNumericalPropertyDef` allows for a limited support of data semantics by means of the following attributes:

`lowerLimit` This value represents a definition of the lower boundary of the allowed interval for this property. The value shall always be provided as a physical value.

`upperLimit` This value represents a definition of the upper boundary of the allowed interval for this property. The value shall always be provided as a physical value.

`unit` This value represents the unit of the property. It is only defied as a simple string without further formalization, i.e. it does not make use of `Unit` and/or `PhysicalDimension`.

`resolution` This attribute defines the resolution of the property. However, this definition should not be confused with a conversion into an internal value domain, comparable to the usage of `CompuMethod`. It just says that the value of the property shall have a certain resolution.

⌋*(RS_MANI_00033)*

For explanation, the values of a REST properties are typically conveyed from sender to receiver on top of a “JSON transport layer”. In other words, the serialization of the values ends up in a string-based format.

There is simply no need to define the conversion into a binary transport format that is used for typical automotive communication buses.

[TPS_MANI_01128]{DRAFT} **Difference between `RestIntegerPropertyDef` and `RestNumberPropertyDef`** ⌈Both `RestIntegerPropertyDef` and `RestNumberPropertyDef` can benefit from the limited support for data semantics as described by [TPS_MANI_01127].

However, by design `RestIntegerPropertyDef` is foreseen to carry integer values while `RestNumberPropertyDef` is reserved for carrying non-integer[1] numbers.⌋ *(RS_MANI_00033)*

Class	***RestAbstractNumericalPropertyDef*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class acts as an abstract base class that contributes attributes for its subclasses that in turn represent a numerical property. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *RestAbstractPropertyDef*, *RestPrimitivePropertyDef*			
Subclasses	RestIntegerPropertyDef, RestNumberPropertyDef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
lowerLimit	Limit	0..1	attr	This attribute specifies the lower limit of the property value. **Tags:**atp.Status=draft
resolution	Float	0..1	attr	This attribute specifies the resolution of a given value on a physical basis. **Tags:**atp.Status=draft
unit	String	0..1	attr	This attribute describes the lower limit of the property's value. **Tags:**atp.Status=draft
upperLimit	Limit	0..1	attr	This attribute describes the upper limit of the property's value. **Tags:**atp.Status=draft

Table 13.17: RestAbstractNumericalPropertyDef

Class	**RestIntegerPropertyDef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to define a REST property with an integer semantics. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *RestAbstractNumericalPropertyDef*, *RestAbstractPropertyDef*, *RestPrimitivePropertyDef*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 13.18: RestIntegerPropertyDef

[1] It would be inaccurate to describe these values as "float" because that would imply a certain representation in a binary layout in memory or on a bus. This binary format is not applicable in this case.

Class	RestNumberPropertyDef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to define a REST property with a numerical semantics. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *RestAbstractNumericalPropertyDef*, *RestAbstractPropertyDef*, *RestPrimitivePropertyDef*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 13.19: RestNumberPropertyDef

[TPS_MANI_01129]{DRAFT} `RestObjectRef` **is only needed for specific implementations of REST-based communication** ⌈The existence of a `RestObjectRef` is only required for specific implementations of the REST-based communication approach.

The application of this reference has some pitfalls (it should only refer to elements in the same service, make sure to only reference the intended kind of element) and therefore needs to be applied carefully.

There is no formal support to make sure that only a certain kind of `RestElementDef` can be referenced. As a semi-formal support for the creation of references the attribute `RestObjectRef.role` has been introduced. It allows for the annotation of the kind of target `RestElementDef`.⌋*(RS_MANI_00033)*

Class	RestObjectRef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to define a REST property that defines reference to another REST element. **Tags:**atp.Status=draft			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *RestAbstractPropertyDef*, *RestPrimitivePropertyDef*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
object	RestElementDef	*	ref	This reference represents the ability to define constraints regarding the reference to another element, i.e. the reference identifies the element to which the reference is allowed to refer. **Tags:**atp.Status=draft
role	String	1	attr	This attribute represents the ability to define a role for the reference to another element. **Tags:**atp.Status=draft

Table 13.20: RestObjectRef

The application of the attribute `RestObjectRef.role` is sketched in Figure 13.7. The example shows a REST service that makes heavy use of the referencing ability.

The roles (in *italics*) can be used for checking, i.e. the reference in the role *engine* should not point to e.g. a gastank object.

But again, this semantics - although the strongest that could be supported on M2 modeling level - is rather weak and may be subject to consistency problems.

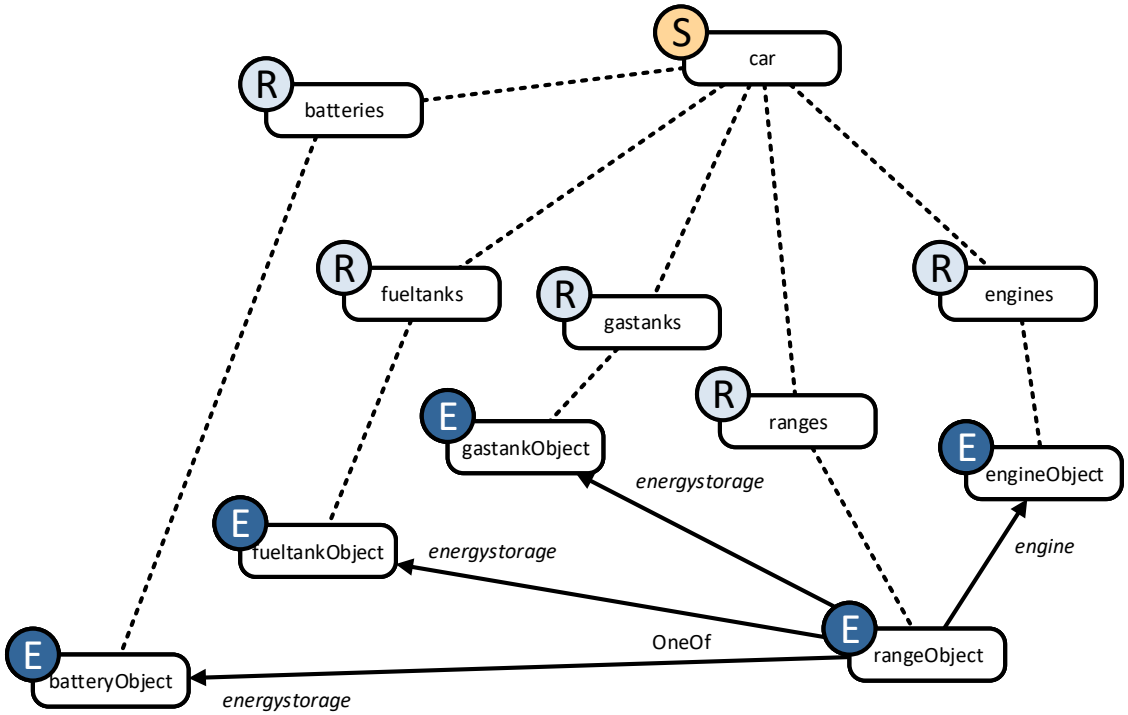


Figure 13.7: Example of the usage of the `role` attribute

13.2 REST Service Deployment

Important note: the AUTOSAR SWS REST [5] defines a low-level `API` for `REST`-based communication. The content of this chapter, on the other hand, applies for the configuration of a not-yet standardized `API` on top of the ara::rest `API`.

The ara::rest `API` requires fully-qualified `URI`s of the *remote communication end* to be passed to the various `API` elements. This is obviously a bad idea if application software should be kept independent of external resources.

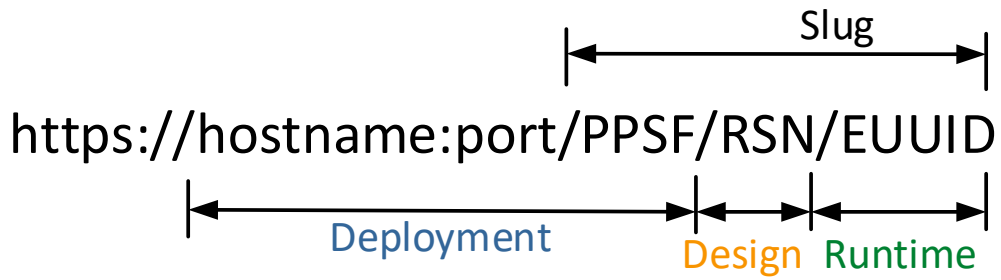
Therefore, an `API` on top of ara::rest could focus on the path of the `URI` that is specific to the respective `REST` service formalized in a `RestServiceInterface` and inject the “non-portable” part of the `URI` of the *remote communication end* within an appropriately configured platform module.

Any approach for this purpose needs to take into account that software can be multiply instantiated (on different levels).

For example, the implementation of an `Executable` shall not make any assumptions about the number and/or behavior of the corresponding `Process`es launched.

This means that the URI may have elements used for the distinction of instances (created by launching the same Executable multiple times according to the definition of Processes in the execution manifest) of the same service.

To further drive this point home, Figure 13.8 has been created as a visualization of how a typical (i.e. it is assumed that RestResourceDef.resource does not exist to keep things simple) REST URI looks like.



Legend
hostname = RestHttpPortPrototypeMapping.host
port = RestHttpPortPrototypeMapping.tcpPort
PPSF = RestHttpPortPrototypeMapping.portPrototypeSlugFragment
RSN = RestResourceDef.shortName
EUUID = UUID of the element assigned at run-time

Figure 13.8: Structure of a typical URI for a REST service

As explained by Figure 13.8, the fully-qualified URI should be composed out of several ingredients contributed by different aspects of the configuration process.

The contribution from the design phase is described in section 13.1. The contribution from the deployment phase is depicted in Figure 13.9.

In addition to the contributions from the design and deployment phase, some information that is only available at run-time when the objects that represents the data of a REST service are allocated in memory makes the list of ingredients for the creation of the URI of a REST service complete.

[TPS_MANI_01130]{DRAFT} **Structure of a typical URI for a REST service** ⌈The part of the URI following the *hostname:port* tuple is usually called the slug.

In the case of a REST service the slug consists of three parts in the order listed below:

1. The representation of the **service instance** (that directly corresponds to the level of a PortPrototype) is contributed by the value of attribute RestHttpPort-PrototypeMapping.portPrototypeSlugFragment. This part is defined on deployment level in order to be sure that it is unique in the context to the *hostname:port* tuple.

2. The **resource** level within the slug is represented by the value of attribute `RestResourceDef.shortName`. This part is contributed on design level.
3. The identification of the **specific element** (on the level of `RestElementDef`) is represented by a `UUID` that is assigned at run-time.

⌋(*RS_MANI_00033*)

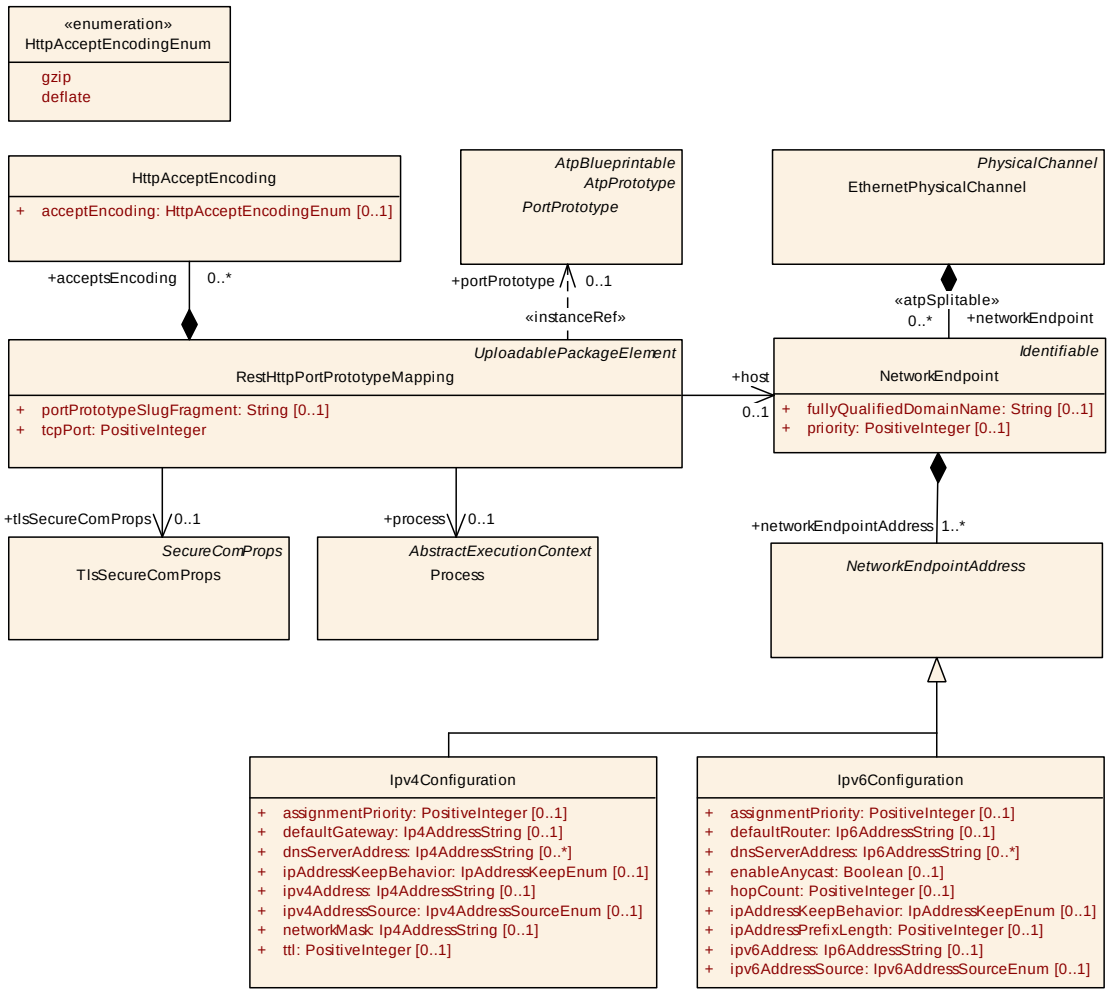


Figure 13.9: Modeling of the `REST` service deployment

In other words, each `URI` represents a specific path within the tree structure rooted in the service level through levels of resources until finally the element level.

While [TPS_MANI_01130] defines the structure for the simplest and probably most like the most popular case (number of resource levels = 1) it is still necessary to understand the impact of more than one resource level on how the `URI` looks like.

This conclusion motivates the existence of [TPS_MANI_01131].

[TPS_MANI_01131]{DRAFT} **Impact of nested `REST` resources on the structure of `REST` URI** ⌈The existence of `RestResourceDef.resource` results in the extension of the design contribution to the `URI` slug by additional levels consisting of the

shortNames of the nested RestResourceDef aggregated in the role resource.
(RS_MANI_00033)

In other words, a specific path through the levels of aggregated RestResourceDefs represented by the respective shortNames, separated by '/' shall be inserted into the "RSN" slot depicted in Figure 13.8.

Please note that the rules for the creation of the slug of a REST URI are more or less arbitrary in terms of the usage of shortName from the model vs. a UUID assigned at run time.

It would technically be possible to use UUIDs instead of shortName on all levels, i.e. also for the "PPSF" and "RSN" slot.

However, this would dramatically decrease the readability of the URI and make it unnecessarily hard for human readers to understand the meaning of a given URI.

Class	**RestHttpPortPrototypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDeployment			
Note	This meta-class represents the ability to define pieces of a URI for the REST service that cannot be contributed from the design point of view. **Tags:** atp.Status=draft atp.recommendedPackage=RestHttpPortPrototypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
accepts Encoding	HttpAcceptEncoding	*	aggr	This aggrgation represents the collection of accepted encodings. **Tags:**atp.Status=draft
host	NetworkEndpoint	0..1	ref	This reference identifies the host configuration of the remote end. **Tags:**atp.Status=draft
portPrototype	PortPrototype	0..1	iref	This reference identifies the instance of the PortPrototype to which the elements of the URI shall be defined. **Tags:**atp.Status=draft **InstanceRef implemented by:**PortPrototypeIn ExecutableInstanceRef
portPrototype SlugFragment	String	0..1	attr	This attribute contributes a string value to be taken as the slug reference that represents the PortPrototype level of a REST service. **Tags:**atp.Status=draft
process	Process	0..1	ref	This reference represents the process required for context of the mapping. **Tags:**atp.Status=draft
tcpPort	PositiveInteger	1	attr	This attribute represents the value of the TCP port applicable for this mapping. **Tags:**atp.Status=draft

▽

△

Class	RestHttpPortPrototypeMapping			
tlsSecureCom Props	TlsSecureComProps	0..1	ref	This represents the configuration of TLS applicable for the mapping. **Tags:**atp.Status=draft

Table 13.21: RestHttpPortPrototypeMapping

[TPS_MANI_01178]{DRAFT} **Semantics of `RestHttpPortPrototypeMapping.acceptsEncoding`** ⌈The aggregation `RestHttpPortPrototypeMapping.acceptsEncoding` allows for a definition of the supported encodings from the client's perspective.

A client may support more than one encoding at the same time. Therefore, the multiplicity of the aggregation has been set to 0..*.⌋*(RS_MANI_00033)*

[constr_1569]{DRAFT} **Restriction for the scope of `RestHttpPortPrototypeMapping.acceptsEncoding`** ⌈The attribute `RestHttpPortPrototypeMapping.acceptsEncoding` shall only be defined on the client side of a communication.⌋ *()*

[constr_1580]{DRAFT} **Restriction for the usage of `RestHttpPortPrototypeMapping.acceptsEncoding`** ⌈Each member of `HttpAcceptEncodingEnum` shall only appear **at most** once in a particular `RestHttpPortPrototypeMapping.acceptsEncoding`.⌋*()*

Please note that a preference rule for one encoding over others in the presence of more than one `RestHttpPortPrototypeMapping.acceptsEncoding` is subject to clarification in the respective SWS [5], see [SWS_REST_01834].

Class	HttpAcceptEncoding			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDeployment			
Note	This meta-class represents the ability to specify the accept-encoding of an exchange using HTTP. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
acceptEncoding	HttpAcceptEncoding Enum	0..1	attr	This attribute is only used on the client side of the configuration for the purpose of stating the accepted compression algorithm.

Table 13.22: HttpAcceptEncoding

Enumeration	HttpAcceptEncodingEnum
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDeployment
Note	This enumeration defines the value for the accept-encoding field of the HTTP header. **Tags:**atp.Status=draft

▽

△

Enumeration	**HttpAcceptEncodingEnum**
Literal	***Description***
deflate	Use deflate compression. **Tags:**atp.EnumerationLiteralIndex=1
gzip	Use gzip pcompression. **Tags:**atp.EnumerationLiteralIndex=0

Table 13.23: HttpAcceptEncodingEnum

14 Software Distribution

14.1 Overview

One of the key features of the *AUTOSAR adaptive platform* is the ability to extend the software on a given ECU without having to re-flash the entire ECU. Instead, software packages are uploaded to the ECU where the content is taken care of by responsible platform modules.

The reason why this topic is relevant for the modeling is the fact that an uploadable software package consists not only of software itself but also of manifest content required to support the integration of the uploaded software with the existing platform instance.

As far as the meta-model is concerned, the discussion about manifests and which manifest content needs to go with which other model elements doesn't care about the file granularity. In other words, it would not make sense to formalize the uploadable software package on the basis of references to files that carry model elements.

Instead, the view on the manifest topic from the modeling point of view focuses on model elements that make up manifest content.

Therefore, the modeling of an uploadable software package allows for putting references to all the required model elements that, in their entirety, make up the manifest of the corresponding application software that is also going to end up in the uploadable software package.

From the formal point of view, such an uploadable software package is modeled as a so-called `SoftwareCluster`. This meta-class is the root element that in turn describes all the necessary content of an uploadable software package.

However, the software package obviously isn't created out of thin air. It is the result of a workflow that starts from the formulation of requirements on the content of a `SoftwareCluster`.

These requirements are formalized by means of meta-class `SoftwareClusterDesign`s.

The relation between `SoftwareClusterDesign` and `SoftwareCluster` is depicted in Figure 14.1.

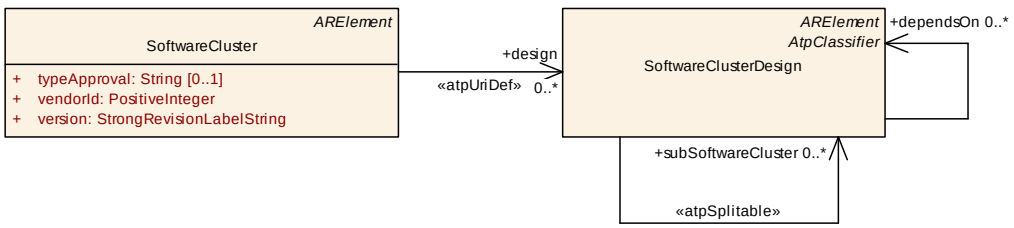


Figure 14.1: Relation of `SoftwareClusterDesign` to `SoftwareCluster`

[TPS_MANI_01109]{DRAFT} **Semantics of `UploadablePackageElement`** ⌈In order to keep the complexity of the modeling of `SoftwareCluster` as low as possible abstract meta-class `UploadablePackageElement` has been created.

This allows for the referencing of model elements derived from `UploadablePackageElement` that need to be considered in an uploadable software package from within a `SoftwareCluster` with just the reference `containedPackageElement`.

The same applies for `SoftwareClusterDesign` and the respective reference `requiredPackageElement`.⌋*(RS_MANI_00035)*

Class	***UploadablePackageElement*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class acts as an abstract base class for all meta-classes that need to be added to an uploadable software package in order to complete the manifest content. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	*AbstractExecutionContext*, *AdaptivePlatformServiceInstance*, CryptoCertificateToPortPrototypeMapping, CryptoKeySlotToPortPrototypeMapping, CryptoProviderToPortPrototypeMapping, DeterministicClient, DltLogChannelToProcessMapping, *FunctionalClusterInteractsWithFunctionalClusterMapping*, PlatformHealthManagementContribution, ProcessExecutionError, ProcessToMachineMappingSet, RawDataStreamDeployment, *RawDataStreamMapping*, RecoveryNotificationToPPortPrototypeMapping, RestHttpPortPrototypeMapping, SecurityEventMapping, *ServiceInstanceToMachineMapping*, ServiceInstanceToPortPrototypeMapping, *ServiceInterfaceDeployment*, StartupConfigSet, TimeSyncPortPrototypeToTimeBaseMapping, *UploadableExclusivePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 14.1: UploadablePackageElement

Please note that this approach to collecting elements is very similar in semantics to `System.fibexElement` or `DiagnosticContributionSet.element`.

14.2 Software Cluster

14.2.1 Software Cluster General Modeling

[TPS_MANI_01110]{DRAFT} **Semantics of `SoftwareCluster`** ⌈The existence of a `SoftwareCluster` represents an uploadable software package.⌋*(RS_MANI_00035)*

[TPS_MANI_01213]{DRAFT} **Semantics of meta-class `StrongRevisionLabelString`** ⌈Meta-class `StrongRevisionLabelString` supports the specification of a version number for a `SoftwareCluster` that consists of four components ([constr_1747] applies):

- Major version
- Minor version
- Patch version

- Additional labels for pre-release version and build metadata

⌋(*RS_MANI_00035*)

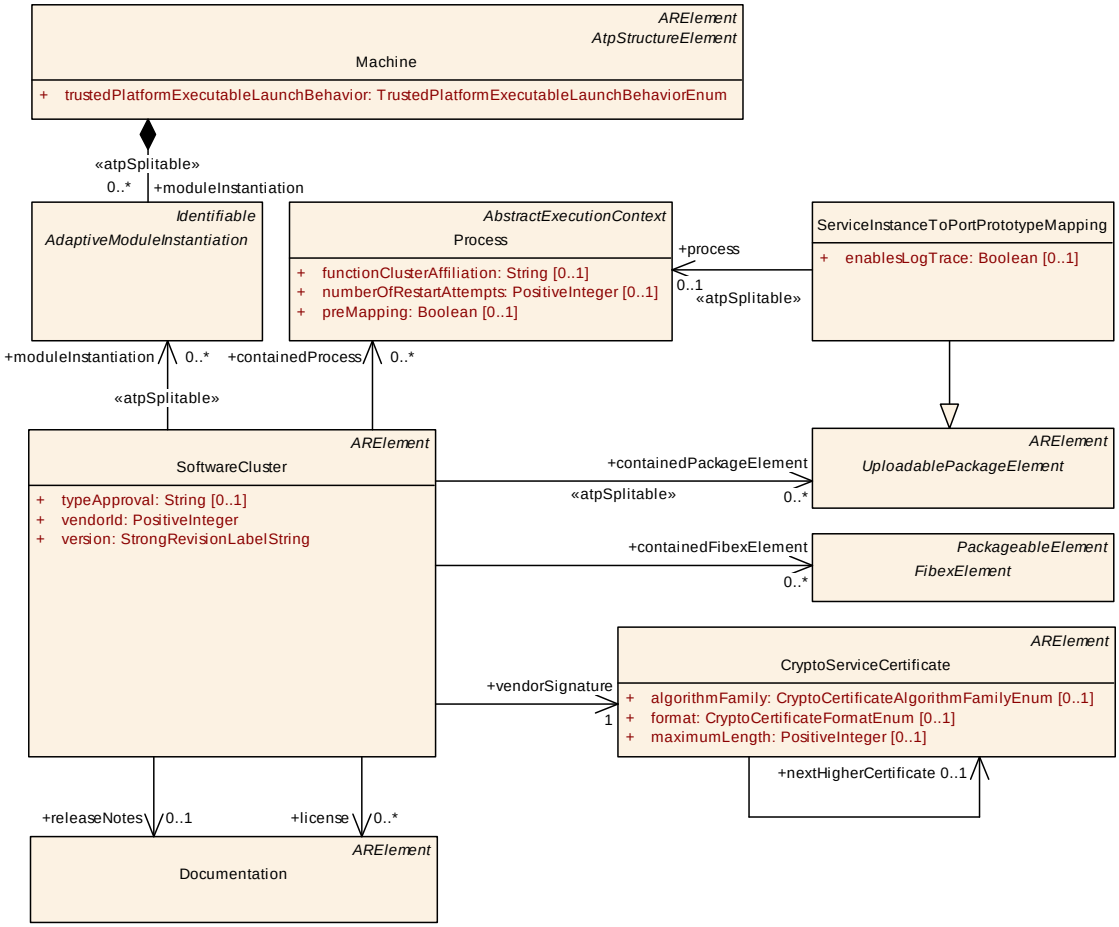


Figure 14.2: Modeling of `SoftwareCluster`

[constr_1747]{DRAFT} **Completeness of the `SoftwareCluster.version`** ⌈The `SoftwareCluster.version` shall contain all the following parts:

- Major version
- Minor version
- Patch version
- Additional labels for pre-release version and build metadata

⌋()

Primitive	**StrongRevisionLabelString**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This primitive represents a revision label which identifies an object under version control. It represents a pattern which requires three integer numbers separated by a dot, representing from left to right Major Version, MinorVersion, PatchVersion and additional labels for pre-release version and build metadata. Legal patterns are for example: 1.0.0-alpha+001 1.0.0+20130313144700 1.0.0-beta+exp.sha.5114f85 **Tags:** atp.Status=draft xml.xsd.customType=STRONG-REVISION-LABEL-STRING xml.xsd.pattern=(0|[1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)(-((0|[1-9]\d*|\d*[a-zA-Z-][0-9a-z A-Z-]*)(\.(0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*))*))?(\+([0-9a-zA-Z-]+(\.[0-9a-zA-Z-]+)*))? xml.xsd.type=string

Table 14.2: StrongRevisionLabelString

Please note that the build number does not necessarily have to be consecutively incremented between two builds. In some cases the build number is created by creating a hash over the build.

In such a case it would not make sense to include the build number in a greater/less comparison while a comparison for equality/inequality may positively make sense. This aspect shall be taken into account when processing the value of an attribute types by a `StrongRevisionLabelString`.

[TPS_MANI_01331]{DRAFT} **Standardized values of attribute `SoftwareCluster.category`** ⌈AUTOSAR standardizes the following values of attribute `SoftwareCluster.category`

- **`PLATFORM_CORE`**: a `SoftwareCluster` of this `category` represents any kind of platform software, e.g. bootloader, hypervisor, OS, adaptive platform module. Such a `SoftwareCluster` cannot be removed by a UCM, but updates are possible.
- **`PLATFORM`**: a `SoftwareCluster` of this `category` represents the parts of the platform software (e.g. configuration of functional clusters) that could be installed, removed, and updated.
- **`APPLICATION_LAYER`**: a `SoftwareCluster` of this `category` represents a driving-relevant function on application level, e.g. a lane keeping assistant, window lift controller, seat positioning. Such a `SoftwareCluster` can be installed, removed, and updated.

⌋()

Please note that it is possible to (in addition to the standardized values) define custom values for attribute `SoftwareCluster.category`.

In this case, however, it is important to use custom values that don't clash with future extensions of the standardized values. A good way to avoid a clash is, for example, to use specific pre- or postfixes that identify a company or project name.

[constr_1788]{DRAFT} **Restriction to `SoftwareCluster` of `category PLATFORM_CORE`** ⌈On each `Machine`, only a single `SoftwareCluster` of `category PLATFORM_CORE` shall be deployed.⌋*()*

[TPS_MANI_01115]{DRAFT} **Specification of executable software within `SoftwareCluster`** ⌈One of the most prominent contents of an uploadable software package is the reference to the executable software.

Within the definition of a `SoftwareCluster`, this reference is implicitly given by means of the reference `SoftwareCluster.containedProcess`.

The target of `SoftwareCluster.containedProcess` is a `Process` that represents an instance of the corresponding executable program (the software image), formalized as `Executable`⌋*(RS_MANI_00035)*

The prominence of the dedicated reference to `Process` is amplified by the fact that it would have been technically possible to let `Process` inherit from `UploadablePackageElement` and thus include the referenced `Process`(es) in the bulk of references to other required model elements.

These references are formalized in two different forms. For technical reasons it is not possible to let all model elements that need to be immediately referenced by a `SoftwareCluster` inherit from `UploadablePackageElement`.

The main reason is that further model elements need to be referenced by a `SoftwareCluster` that are also used on the *AUTOSAR classic platform*.

In other words, it would be very questionable to introduce the “useless” concept of an `UploadablePackageElement` into the scope of the *AUTOSAR classic platform* as a mere (and unwanted) side effect of providing a definition of the `SoftwareCluster` on the *AUTOSAR classic platform*.

The scope of a single `SoftwareCluster` in terms of a relation to a `Machine` is that all software contained in one `SoftwareCluster` is supposed to be uploaded to one and only one `Machine`.

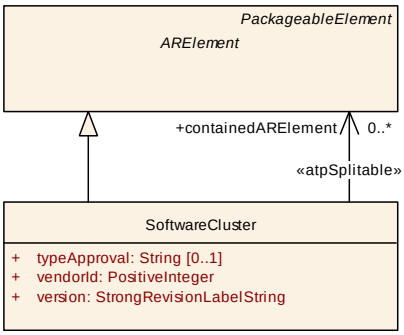


Figure 14.3: `SoftwareCluster` can reference `ARElement`

The definition of `SoftwareCluster` shall never include multiple `Machine`s. This is expressed in [constr_1536].

[constr_1536]{DRAFT} **Definition of `SoftwareCluster` applies for a single `Machine`** ⌈Within the scope of a `SoftwareCluster`, each `Process` referenced in the role `containedProcess` shall be mapped (e.g. by means of the existence of a `ProcessToMachineMapping`) to the same `Machine`.⌋*()*

[TPS_MANI_01116]{DRAFT} **Reference to model elements included in an uploadable software package** ⌈Beside the ability to explicitly reference a `Process` in the role `containedProcess` it is possible to define the following references to required model elements:

- references to meta-classes derived from `UploadablePackageElement` are formalized by way of `SoftwareCluster.containedPackageElement`.
- references to meta-classes derived from `ARElement` are formalized by way of `SoftwareCluster.containedARElement`.
- references to meta-classes derived from `FibexElement` are formalized by way of `SoftwareCluster.containedFibexElement`.

Technically, an `UploadablePackageElement` is also an `ARElement`, but it is still mandated to use the dedicated reference specifically for `UploadablePackageElement`.⌋*(RS_MANI_00035)*

To exemplify the reference to `UploadablePackageElement`, Figure 14.2 contains a subclass of `UploadablePackageElement`: `ServiceInstanceToPortPrototypeMapping`.

It is obvious that the uploaded software needs to integrate with the communication stack and `ServiceInstanceToPortPrototypeMapping` is a prominent model element for this purpose.

[TPS_MANI_01202]{DRAFT} **Semantics of reference `SoftwareCluster.moduleInstantiation`** ⌈By means of the reference `SoftwareCluster.moduleInstantiation` it is possible to express the need for updates of the platform infrastructure along with other resources referenced by the enclosing `SoftwareCluster`.⌋ *(RS_MANI_00035)*

[TPS_MANI_01218]{DRAFT} **Cryptographic signature of `SoftwareCluster`** ⌈A `SoftwareCluster` also needs to be signed cryptographically. For this purpose, meta-class `CryptoServiceCertificate` is referenced in the role `vendorSignature`.⌋ *(RS_MANI_00035)*

[TPS_MANI_01219]{DRAFT} **License of software in included `SoftwareCluster`** ⌈It is possible to refer to licenses for software included in a `SoftwareCluster` by means of a reference to meta-class `Documentation` in the role `license`.⌋*(RS_MANI_00035)*

Class	**Documentation**			
Package	M2::AUTOSARTemplates::GenericStructure::DocumentationOnM1			
Note	This meta-class represents the ability to handle a so called standalone documentation. Standalone means, that such a documentation is not embedded in another ARElement or identifiable object. The standalone documentation is an entity of its own which denotes its context by reference to other objects and instances. **Tags:**atp.recommendedPackage=Documentations			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
context	DocumentationContext	*	aggr	This is the context of the particular documentation.
documentation Content	PredefinedChapter	0..1	aggr	This is the content of the documentation related to the specified contexts. **Tags:**xml.sequenceOffset=200

Table 14.3: Documentation

Please note that `Documentation` is an `ARElement` that cannot be owned by a `SoftwareCluster`. The latter can only refer to it.

This aspect also means that once a given `license` is formalized by means of a `Documentation` it is in general possible to refer to this formalization from within different `SoftwareCluster`s.

[TPS_MANI_01220]{DRAFT} **Release notes of software in included `SoftwareCluster`** ⌈It is possible to refer to release notes for software included in a `SoftwareCluster` by means of a reference to meta-class `Documentation` in the role `releaseNotes`.⌋*(RS_MANI_00035)*

[constr_1566]{DRAFT} **Usage of `SoftwareCluster.containedARElement`** ⌈The reference `SoftwareCluster.containedARElement` shall not be used to refer to a `SoftwareCluster` or a `SoftwareClusterDesign`.⌋*()*

14.2.2 Relevance of Software Cluster for Diagnostics

[TPS_MANI_01111]{DRAFT} **Diagnostic Address of a `SoftwareCluster`** ⌈An uploadable software package formalized as a `SoftwareCluster` will typically be equipped with a diagnostics management component.

Therefore, the definition of the `SoftwareCluster` needs to provide information about the diagnostic address(es) to which the contained diagnostic management component shall respond.

This information is formalized by means of the attribute `SoftwareCluster.diagnosticAddress`.

A `SoftwareCluster` may be required to respond to multiple (i.e. several functional plus one physical) diagnostic addresses, thus the multiplicity of `diagnosticAddress` is set to 0..*.⌋*(RS_MANI_00035)*

Please note that the modeling of the SoftwareClusterDiagnosticAddress has been created with the primary goal to support the usage of DoIP for diagnostics.

The secondary goal has been to make the modeling of the diagnostic address extensible such that the idiomatic ways in which other transport layers (CAN, LIN, FlexRay, etc.) define diagnostic addresses can also be supported by adding respective subclasses of SoftwareClusterDiagnosticAddress.

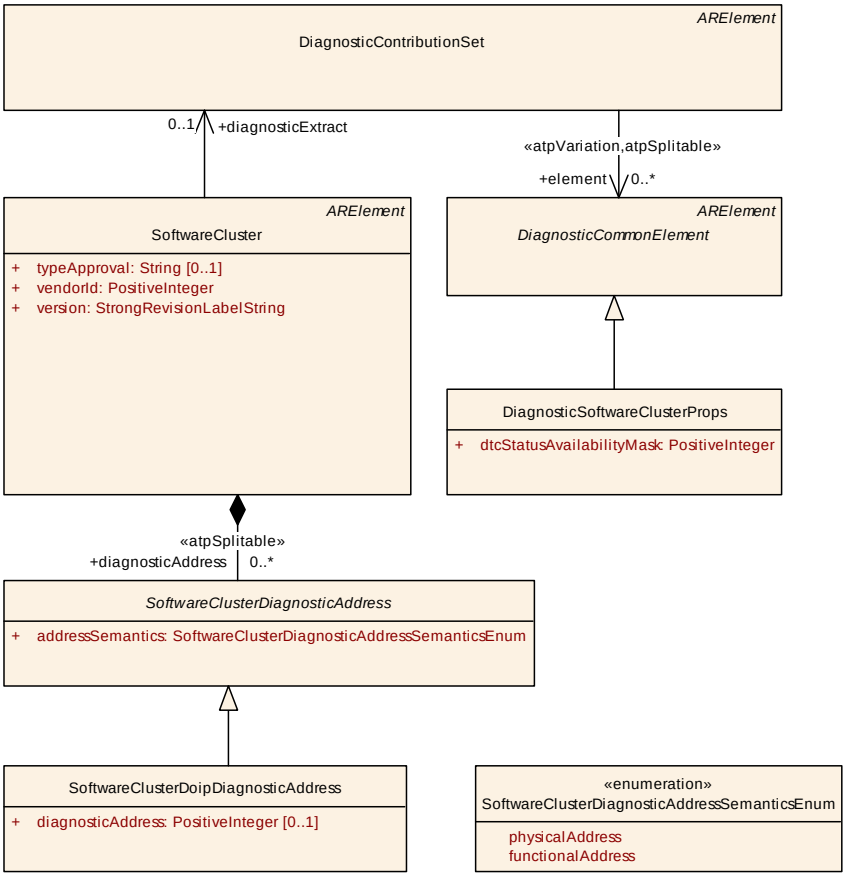


Figure 14.4: Modeling of diagnostic address of a SoftwareCluster

[constr_1543]{DRAFT} **Only one physical address per SoftwareCluster** ⌈Each SoftwareCluster shall only aggregate one SoftwareClusterDiagnosticAddress where the value of attribute addressSemantics is set to SoftwareClusterDiagnosticAddressSemanticsEnum.physicalAddress.⌋*()*

Class	**SoftwareCluster**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This meta-class represents the ability to define an uploadable software-package, i.e. the SoftwareCluster shall contain all software and configuration for a given purpose. **Tags:** atp.Status=draft atp.recommendedPackage=SoftwareClusters

▽

△

Class	**SoftwareCluster**			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
claimed FunctionGroup	ModeDeclarationGroup Prototype	*	ref	Each SoftwareCluster can reserve the usage of a given functionGroup such that no other SoftwareCluster is allowed to use it **Tags:**atp.Status=draft
conflictsTo	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation handles conflicts. If it yields true then the SoftwareCluster shall not be installed. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=conflictsTo atp.Status=draft
contained ARElement	ARElement	*	ref	This reference represents the collection of model elements that cannot derive from UploadablePackage Element and that contribute to the completeness of the definition of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=containedARElement atp.Status=draft
containedFibex Element	FibexElement	*	ref	This allows for referencing FibexElements that need to be considered in the context of a SoftwareCluster. **Tags:**atp.Status=draft
contained Package Element	UploadablePackage Element	*	ref	This reference identifies model elements that are required to complete the manifest content. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=containedPackageElement atp.Status=draft
contained Process	Process	*	ref	This reference represent the processes contained in the enclosing SoftwareCluster. **Tags:**atp.Status=draft
dependsOn	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation can be taken to identify a dependency for the enclosing SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=dependsOn atp.Status=draft
design	SoftwareClusterDesign	*	ref	This reference represents the identification of all Software ClusterDesigns applicable for the enclosing Software Cluster. **Stereotypes:** atpUriDef **Tags:**atp.Status=draft
diagnostic Address	SoftwareCluster DiagnosticAddress	*	aggr	This aggregation represents the collection of diagnostic addresses that apply for the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=diagnosticAddress atp.Status=draft
diagnostic Extract	DiagnosticContribution Set	0..1	ref	This reference represents the definition of the diagnostic extract applicable to the referencing SoftwareCluster **Tags:**atp.Status=draft

▽

△

Class	**SoftwareCluster**			
license	Documentation	*	ref	This attribute allows for the inclusion of the the full text of a license of the enclosing SoftwareCluster. In many cases open source licenses require the inclusion of the full license text to any software that is released under the respective license. **Tags:**atp.Status=draft
module Instantiation	AdaptiveModule Instantiation	*	ref	This reference identifies AdaptiveModuleInstantiations that need to be included with the SoftwareCluster in order to establish infrastructure required for the installation of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=moduleInstantiation atp.Status=draft
releaseNotes	Documentation	0..1	ref	This attribute allows for the explanations of changes since the previous version. The list of changes might require the creation of multiple paragraphs of test. **Tags:**atp.Status=draft
typeApproval	String	0..1	attr	This attribute carries the homologation information that may be specific for a given country.
vendorId	PositiveInteger	1	attr	Vendor ID of this Implementation according to the AUTOSAR vendor list.
vendor Signature	CryptoService Certificate	1	ref	This reference identifies the certificate that represents the vendor's signature. **Tags:**atp.Status=draft
version	StrongRevisionLabel String	1	attr	This attribute can be used to describe a version information for the enclosing SoftwareCluster.

Table 14.4: SoftwareCluster

Class	***SoftwareClusterDiagnosticAddress*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define a diagnostic address in an abstract form. Sub-classes are supposed to clarify how the diagnostic address shall be defined according to the applicable addressing scheme (DoIP vs. CAN TP vs. ...). **Tags:**atp.Status=draft			
Base	*ARObject*			
Subclasses	SoftwareClusterDoipDiagnosticAddress			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
address Semantics	SoftwareCluster DiagnosticAddress SemanticsEnum	1	attr	This attribute clarifies whether the address value shall be interpreted as a physical or a functional address.

Table 14.5: SoftwareClusterDiagnosticAddress

Enumeration	**SoftwareClusterDiagnosticAddressSemanticsEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution

▽

△

Enumeration	**SoftwareClusterDiagnosticAddressSemanticsEnum**
Note	This meta-class defines a list of semantics for the interpretation of diagnostic addresses in the context of a SoftwareCluster. **Tags:**atp.Status=draft
Literal	***Description***
functionalAddress	This address represents a functional address. **Tags:**atp.EnumerationLiteralIndex=1
physicalAddress	This address represents a physical address. **Tags:**atp.EnumerationLiteralIndex=0

Table 14.6: SoftwareClusterDiagnosticAddressSemanticsEnum

Class	**SoftwareClusterDoipDiagnosticAddress**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define a diagnostic address specifically for the DoIP case. **Tags:**atp.Status=draft			
Base	*ARObject*, *SoftwareClusterDiagnosticAddress*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnostic Address	PositiveInteger	0..1	attr	This attribute represents the collection of diagnostic addresses the SoftwareCluster occupies. **Tags:**atp.Status=draft

Table 14.7: SoftwareClusterDoipDiagnosticAddress

[TPS_MANI_01114]{DRAFT} **Relation of `DiagnosticContributionSet` to `SoftwareCluster`** ⌈In AUTOSAR, the formalization of the external behavior of the diagnostic stack is rooted in meta-class `DiagnosticContributionSet`.

On the *AUTOSAR classic platform* the scope of the “external behavior of the diagnostic stack” is represented by an entire ECU.

This relation changes on the *AUTOSAR adaptive platform* where each uploadable software package is shipped with the definition of the “external behavior of the diagnostic stack” **as far as the software in the scope of respective uploadable software package is concerned**.

To fully support the different approaches of *AUTOSAR classic platform* and *AUTOSAR adaptive platform* it is necessary to provide means for specifying a `DiagnosticContributionSet` for a given `SoftwareCluster`.

In particular, this relation is created by means of the reference `SoftwareCluster.diagnosticExtract`.⌋*(RS_MANI_00035)*

In other words, the “external behavior of the diagnostic stack” of each `SoftwareCluster` shall only be described by a single `DiagnosticContributionSet`.

And since the `DiagnosticContributionSet` and all referenced `element`s are subject to the upload on a target platform it only makes sense that the `SoftwareCluster` references the `DiagnosticContributionSet` (instead of the other way round).

[constr_1568]{DRAFT} **Existence of `SoftwareCluster.diagnosticExtract`** ⌈The reference `SoftwareCluster.diagnosticExtract` shall only exist if the `SoftwareCluster` is not referenced by in the role `SoftwareClusterDependencyCompareCondition.softwareCluster` by a `SoftwareClusterDependencyCompareCondition` that is eventually aggregated by a `SoftwareClusterDependencyFormula` of `category STRUCTURAL_DEPENDENCY`.⌋*()*

Rationale for the existence of [constr_1562]: the definition of the diagnostic behavior is limited to the root level of a structure of `SoftwareCluster`s in the same spirit that caused the existence of [constr_1564].

Class	**DiagnosticSoftwareClusterProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to specify properties for the relation between a Diagnostic ContributionSet and a SoftwareCluster. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticSoftwareClusterPropss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dtcStatus AvailabilityMask	PositiveInteger	1	attr	This attribute contains the value of the DTC status availability mask. **Tags:**atp.Status=draft

Table 14.8: DiagnosticSoftwareClusterProps

[constr_1535]{DRAFT} **Existence of `DiagnosticSoftwareClusterProps` in the context of a `DiagnosticContributionSet`** ⌈Each `DiagnosticContributionSet` shall only reference a single `DiagnosticSoftwareClusterProps` in the role `element`.⌋*()*

[constr_1564]{DRAFT} **Existence of `SoftwareCluster.diagnosticAddress`** ⌈The aggregation of `SoftwareClusterDiagnosticAddress` at `SoftwareCluster` in the role `diagnosticAddress` shall only exist if the `SoftwareCluster` is not referenced by in the role `SoftwareClusterDependencyCompareCondition.softwareCluster` by a `SoftwareClusterDependencyCompareCondition` that is eventually aggregated by a `SoftwareClusterDependencyFormula` of `category STRUCTURAL_DEPENDENCY`.⌋*()*

Please note that the consequence of [constr_1564] is that a `SoftwareCluster` to which **no structural dependency** is defined represents a root software cluster. A diagnostic address can only be owned by a root software cluster.

14.2.3 Sub Software Cluster

It is possible to logically subdivide a `SoftwareCluster` into smaller parts. The motivation for breaking down a `SoftwareCluster` might be that parts of the development are subcontracted to other organizations.

The formalization of the subdivision of SoftwareClusters is done by means of the definition of the specification of dependency among SoftwareClusters. The details are explained in chapter 14.2.4.

14.2.4 Software Cluster Dependency

[TPS_MANI_01215]{DRAFT} **Semantics of meta-class SoftwareClusterDependencyFormula** ⌈Meta-class SoftwareClusterDependencyFormula allows for the definition of a formal condition that can be taken to decide about the dependency to or the conflict with a SoftwareCluster.

The modeling of SoftwareClusterDependencyFormula allows for the definition of nested conditions. The attribute operator is applied on the results of the evaluation of the parts.⌋*(RS_MANI_00035)*

Class	**SoftwareClusterDependencyFormula**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define a dependency among SoftwareClusters. **Tags:**atp.Status=draft			
Base	*ARObject*, *SoftwareClusterDependencyFormulaPart*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
category	CategoryString	0..1	attr	This attribute specializes the semantics of the enclosing SoftwareClusterDependencyFormula.
operator	SoftwareCluster DependencyLogical OperatorEnum	0..1	attr	This logical operator can be used to relate the results of different SoftwareClusterDependencyParts.
part (ordered)	SoftwareCluster DependencyFormula Part	*	aggr	This aggregation represents the ordered collection of the parts of the SoftwareClusterDependencyFormula. **Tags:**atp.Status=draft

Table 14.9: SoftwareClusterDependencyFormula

Enumeration	**SoftwareClusterDependencyLogicalOperatorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This enumeration provides a set of operators to be used in a SoftwareClusterDependencyFormula. **Tags:**atp.Status=draft
Literal	***Description***
logicalAnd	logical and **Tags:**atp.EnumerationLiteralIndex=0
logicalOr	logical or **Tags:**atp.EnumerationLiteralIndex=1

Table 14.10: SoftwareClusterDependencyLogicalOperatorEnum

[TPS_MANI_01335]{DRAFT} **Semantics of SoftwareClusterDependencyFormula.category** ⌈The following values of attribute SoftwareClusterDependencyFormula.category are standardized by AUTOSAR:

FUNCTIONAL_DEPENDENCY The role of the SoftwareClusterDependencyFormula is to implement a **functional dependency**, i.e. the SoftwareCluster on the top of the aggregation chain in the role dependsOn relies on the target of SoftwareClusterDependencyCompareCondition.softwareCluster.

STRUCTURAL_DEPENDENCY The SoftwareCluster on the top of the aggregation chain in the role dependsOn is considered a root SoftwareCluster and the SoftwareClusters referenced in the role SoftwareClusterDependencyCompareCondition.softwareCluster represent sub SoftwareClusters.

⌋(*RS_MANI_00035*)

[TPS_MANI_01216]{DRAFT} **Semantics of meta-class SoftwareClusterDependencyFormulaPart** ⌈Meta-class SoftwareClusterDependencyFormulaPart represents a part of a SoftwareClusterDependencyFormula. The order of the parts of a SoftwareClusterDependencyFormula is significant.⌋(*RS_MANI_00035*)

Class	*SoftwareClusterDependencyFormulaPart* (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents an abstract base class for the definition of different formula parts of a SoftwareClusterDependencyFormula. **Tags:**atp.Status=draft			
Base	*ARObject*			
Subclasses	SoftwareClusterDependencyCompareCondition, SoftwareClusterDependencyFormula			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 14.11: SoftwareClusterDependencyFormulaPart

At the same time, SoftwareClusterDependencyFormulaPart is the base class of SoftwareClusterDependencyFormula.

This means that the SoftwareClusterDependencyFormula can aggregate all subclasses of SoftwareClusterDependencyFormulaPart, i.e. SoftwareClusterDependencyFormula and SoftwareClusterDependencyCompareCondition.

[TPS_MANI_01164]{DRAFT} **Semantics of SoftwareCluster.dependsOn** ⌈A SoftwareCluster has the ability to express a dependency to other SoftwareClusters in the role dependsOn. The semantics of this aggregation depends itself on the value of attribute SoftwareClusterDependencyFormula.category.

Attribute SoftwareCluster.dependsOn allows for the definition of a **formal** (potentially nested) dependency condition. The dependency shall be applicable only if the condition defined by dependsOn yields True.⌋(*RS_MANI_00035*)

[TPS_MANI_01217]{DRAFT} **Semantics of meta-class SoftwareClusterDependencyCompareCondition** ⌈Meta-class SoftwareClusterDependencyCompareCondition allows for the definition of a formal condition to compare against the version of the referenced softwareCluster using a given compareType.

The ability to specifically decide about whether to consider the build number in the comparison is implemented by means of attribute considerBuildNumber.⌋(*RS_-MANI_00035*)

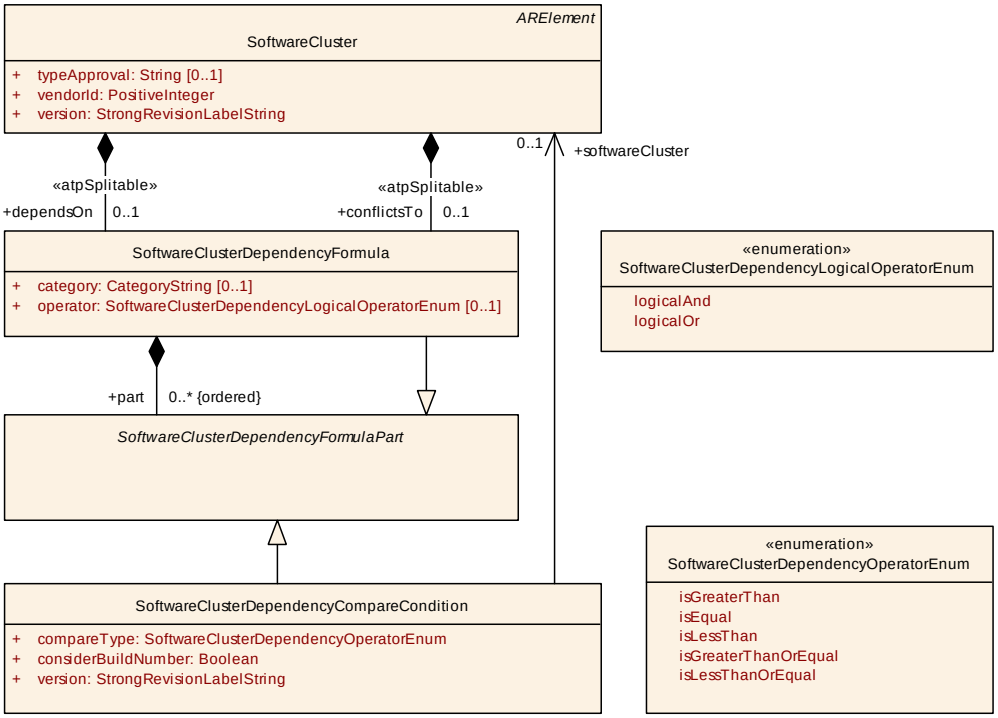


Figure 14.5: Modeling of dependencies in the context of a SoftwareCluster and SoftwareClusterDesign

Class	**SoftwareClusterDependencyCompareCondition**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to specify a concrete dependency condition in the context of a SoftwareClusterDependencyFormula. **Tags:**atp.Status=draft			
Base	*ARObject*, *SoftwareClusterDependencyFormulaPart*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compareType	SoftwareCluster DependencyOperator Enum	1	attr	This attribute identifies the semantics of the compare operator.
considerBuild Number	Boolean	1	attr	If this attribute is set to true then the build number shall be taken into account for the comparison. Build numbers don't have to be consecutive but could be created by some kind of hashing algorithm. In such a case it might make sense to include the build number in a test for equality but it is probably not reasonable to apply e.g. a less-than comparison.
softwareCluster	SoftwareCluster	0..1	ref	This reference identifies the SoftwareCluster to which the dependency/conflict applies. **Tags:**atp.Status=draft

△

Class	**SoftwareClusterDependencyCompareCondition**			
version	StrongRevisionLabel String	1	attr	This attribute represents the value of a version against which the comparison shall be executed.

Table 14.12: SoftwareClusterDependencyCompareCondition

Enumeration	**SoftwareClusterDependencyOperatorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This enumeration provides a choice of operators for comparison within a SoftwareCluster DependencyCompareCondition. **Tags:**atp.Status=draft
Literal	***Description***
isEqual	equal **Tags:**atp.EnumerationLiteralIndex=1
isGreaterThan	greater than **Tags:**atp.EnumerationLiteralIndex=0
isGreaterThanOr Equal	greater than or equal **Tags:**atp.EnumerationLiteralIndex=3
isLessThan	less than **Tags:**atp.EnumerationLiteralIndex=2
isLessThanOrEqual	less than or equal **Tags:**atp.EnumerationLiteralIndex=4

Table 14.13: SoftwareClusterDependencyOperatorEnum

This relation is exemplified by the following sketch (where the orange ellipsis represent `SoftwareClusterDependencyFormula` and the blue ellipsis represent `SoftwareClusterDependencyCompareCondition`) and a corresponding ARXML formalization:

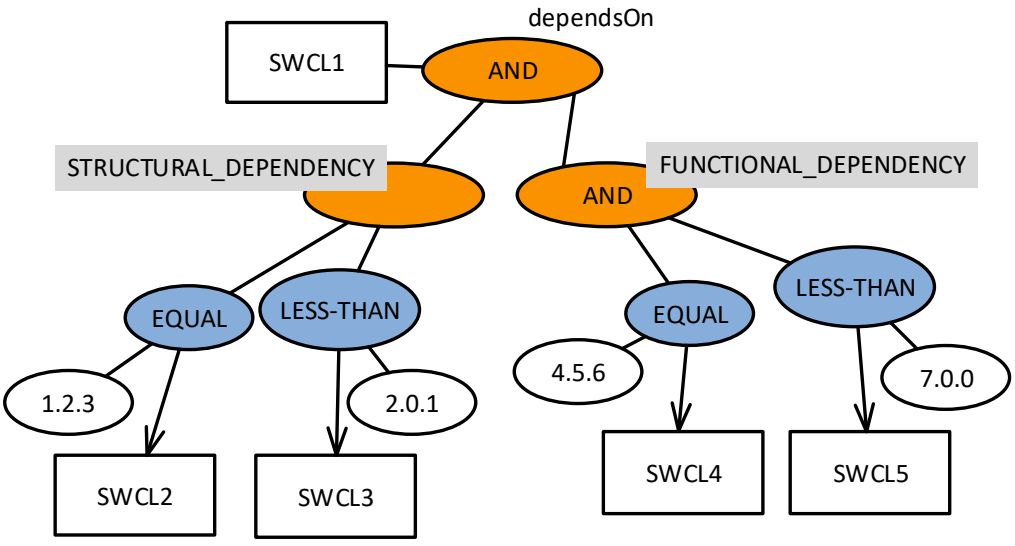


Figure 14.6: Example how dependencies among `SoftwareCluster`s can be defined.

```
<SOFTWARE-CLUSTER>
  <SHORT-NAME>SWCL1</SHORT-NAME>
  <DEPENDS-ON>
    <PARTS>
      <SOFTWARE-CLUSTER-DEPENDENCY-FORMULA>
        <OPERATOR>LOGICAL-AND</OPERATOR>
        <PARTS>
          <SOFTWARE-CLUSTER-DEPENDENCY-FORMULA>
            <CATEGORY>STRUCTURAL_DEPENDENCY</CATEGORY>
            <PARTS>
              <SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
                <COMPARE-TYPE>IS-EQUAL</COMPARE-TYPE>
                <SOFTWARE-CLUSTER-REF DEST="SOFTWARE-CLUSTER">/
    SoftwareClusters/SWCL2</SOFTWARE-CLUSTER-REF>
                <VERSION>1.2.3</VERSION>
              </SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
              <SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
                <COMPARE-TYPE>IS-GREATER-THAN</COMPARE-TYPE>
                <SOFTWARE-CLUSTER-REF DEST="SOFTWARE-CLUSTER">/
    SoftwareClusters/SWCL3</SOFTWARE-CLUSTER-REF>
                <VERSION>2.0.1</VERSION>
              </SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
            </PARTS>
          </SOFTWARE-CLUSTER-DEPENDENCY-FORMULA>
          <SOFTWARE-CLUSTER-DEPENDENCY-FORMULA>
            <CATEGORY>FUNCTIONAL_DEPENDENCY</CATEGORY>
            <OPERATOR>LOGICAL-AND</OPERATOR>
            <PARTS>
              <SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
                <COMPARE-TYPE>IS-EQUAL</COMPARE-TYPE>
                <SOFTWARE-CLUSTER-REF DEST="SOFTWARE-CLUSTER">/
    SoftwareClusters/SWCL4</SOFTWARE-CLUSTER-REF>
                <VERSION>4.5.6</VERSION>
              </SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
              <SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
                <COMPARE-TYPE>IS-LESS-THAN</COMPARE-TYPE>
                <SOFTWARE-CLUSTER-REF DEST="SOFTWARE-CLUSTER">/
    SoftwareClusters/SWCL5</SOFTWARE-CLUSTER-REF>
                <VERSION>7.0.0</VERSION>
              </SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
            </PARTS>
          </SOFTWARE-CLUSTER-DEPENDENCY-FORMULA>
        </PARTS>
      </SOFTWARE-CLUSTER-DEPENDENCY-FORMULA>
    </PARTS>
  </DEPENDS-ON>
</SOFTWARE-CLUSTER>
```

Listing 14.1: Example for the definition of a dependency between SoftwareClusters

Please note (as depicted in the example) that this modeling allows for the definition of a **functional dependency** and a **structural dependency** at the same time.

This configuration is possible by aggregating a SoftwareClusterDependencyFormula that does not define a value for attribute category but immediately aggregates:

- a SoftwareClusterDependencyFormula where attribute category is set to the value STRUCTURAL_DEPENDENCY
- a SoftwareClusterDependencyFormula where attribute category is set to the value FUNCTIONAL_DEPENDENCY

[constr_10010]{DRAFT} **Usage of attribute category in a SoftwareClusterDependencyFormula** ⌈Within an aggregation of (a chain of) SoftwareClusterDependencyFormula the values STRUCTURAL_DEPENDENCY and FUNCTIONAL_DEPENDENCY shall only be used once (preferably on the top-most or second nesting level).⌋ *()*

The existence of sub-software-clusters is restricted to one level, i.e. a sub-software-cluster may not define further sub-software-clusters. This restriction is formalized in the existence of [constr_10011].

[constr_10011]{DRAFT} **Definition of sub-software-cluster** ⌈A SoftwareCluster that is referenced in the role SoftwareClusterDependencyCompareCondition.softwareCluster by a SoftwareClusterDependencyCompareCondition that is eventually aggregated by a SoftwareClusterDependencyFormula of category STRUCTURAL_DEPENDENCY shall not itself aggregate (on any level) a SoftwareClusterDependencyFormula of category STRUCTURAL_DEPENDENCY.⌋*()*

[TPS_MANI_01214]{DRAFT} **Semantics of SoftwareCluster.conflictsTo** ⌈A SoftwareCluster has the ability to express a conflict to other SoftwareClusters in the role conflictsTo. The semantics is to express that the functionality of the referenced SoftwareCluster **inhibits** the installation of the referencing SoftwareCluster.

Attribute SoftwareCluster.conflictsTo allows for the definition of a **formal** (potentially nested) dependency condition. The dependency shall be applicable only if the condition defined by conflictsTo yields False.⌋*(RS_MANI_00035)*

14.2.5 References between Software Clusters

There are several strong use cases for the need of referencing into different SoftwareClusters, for example:

- Reference to a ProvidedApServiceInstance of RequiredApServiceInstance defined in the context of a "host" SoftwareCluster.
- Reference to CommunicationConnectors defined on Machine level from within application SoftwareClusters.
- Reference from a Process in one SoftwareCluster to an Executable in another SoftwareCluster with the semantics that the referencing Process is just another instance of the Executable.

To support such use cases, AUTOSAR provides the definition of dependencies among SoftwareClusters such that a SoftwareCluster that contains a reference can define a dependency to another SoftwareCluster that contains the referenced object.

[TPS_MANI_01329]{DRAFT} **Reference to model elements in different SoftwareClusters** ⌈If a model element inside a given SoftwareCluster defines a reference to another model element and the referenced model element is contained in a different SoftwareCluster, then the SoftwareCluster that contains the referencing model element shall establish a dependency to the other SoftwareCluster by means of an aggregation of SoftwareClusterDependencyFormula in the role dependsOn.⌋*()*

[constr_1784]{DRAFT} **Restriction for the reference to UploadableExclusivePackageElement** ⌈A reference to an UploadableExclusivePackageElement shall not cross the boundary of the enclosing SoftwareCluster, i.e. the target UploadableExclusivePackageElement of such a reference shall not be located in a different SoftwareCluster than the owner of the reference.⌋*()*

Note that [constr_1784] forbids a reference across SoftwareClusters to an UploadableExclusivePackageElement, regardless of whether there is a dependency relation defined or not.

Referencing from one SoftwareCluster into another SoftwareCluster is only allowed if

- the referenced SoftwareCluster is in the list of dependent SoftwareClusters for the referencing SoftwareCluster and the value of attribute SoftwareClusterDependencyFormula.category is set to FUNCTIONAL_DEPENDENCY.
- between referencing and referenced SoftwareCluster a dependency exists where the value of SoftwareClusterDependencyFormula is set to STRUCTURAL_DEPENDENCY. This means that in the relation between a root-SoftwareCluster and a sub-SoftwareCluster references from "root to sub" and from" sub to root" are possible.

This restriction is formalized in [constr_1785].

[constr_1785]{DRAFT} **Restriction regarding the reference into another SoftwareCluster** ⌈A reference from an element in one SoftwareCluster to an element located in another SoftwareCluster shall only exist if

- the SoftwareCluster that owns the referencing element aggregates a SoftwareClusterDependencyFormula of category FUNCTIONAL_DEPENDENCY in the role dependsOn and the SoftwareCluster that owns the referenced element is referenced by a SoftwareClusterDependencyCompareCondition in the context of the mentioned SoftwareClusterDependencyFormula in the role part.softwareCluster or

- one (either referencing of referenced) of the SoftwareClusters owns a SoftwareClusterDependencyFormula of category STRUCTURAL_DEPENDENCY and the other (either referenced or referencing) SoftwareCluster is referenced in the role softwareCluster by a SoftwareClusterDependencyCompareCondition in the context of the enclosing SoftwareClusterDependencyFormula.

For these cases, [constr_1784] applies.⌋()

Please note that the motivation for creating a constellation of one root-SoftwareCluster and one or more sub-SoftwareCluster is entirely driven by development-logistics, i.e. the development of one SoftwareCluster is partly subcontracted to a different company.

From the semantical perspective, the root-SoftwareCluster may need to access elements of the sub-SoftwareCluster and vice versa.

14.3 Software Package

The existence of the SoftwareCluster by itself is not sufficient for installation. Actually, the SoftwareCluster gets wrapped into a so-called SoftwarePackage that comes with an own manifest format that is at least partly standardized.

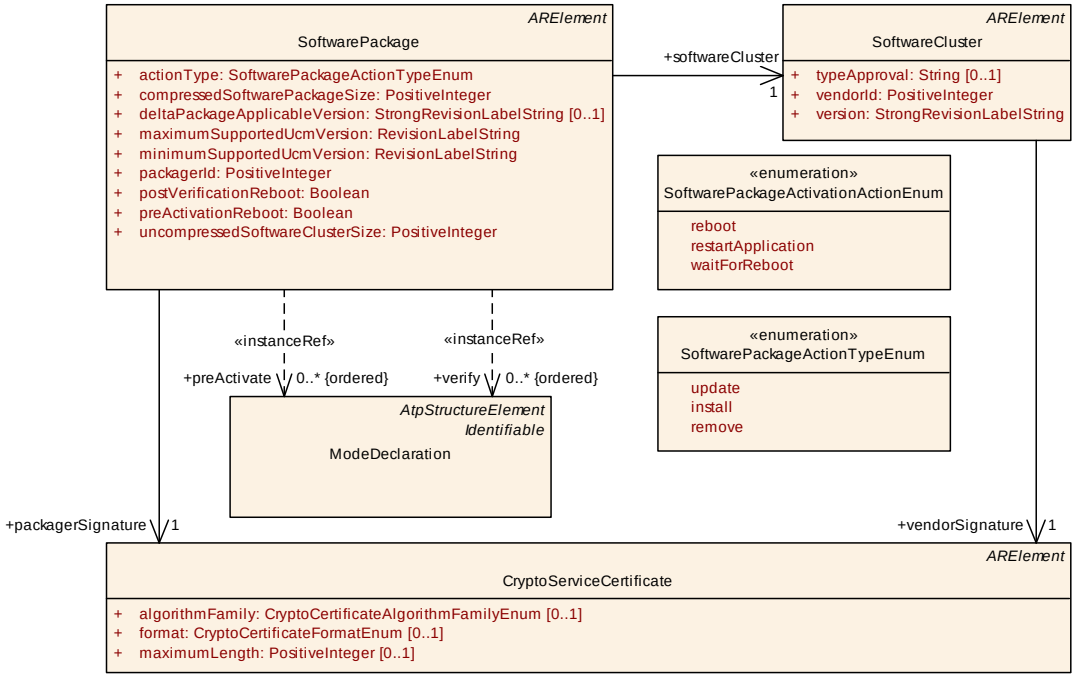


Figure 14.7: Modeling of SoftwarePackage

The difference between the semantics of a SoftwareCluster and the semantics of SoftwarePackage is that a SoftwareCluster focusses on the structure of the software itself while the SoftwarePackage is created to handle the logistics aspect of the software installation.

[TPS_MANI_01221]{DRAFT} **Semantics of meta-class SoftwarePackage** ⌈The purpose of meta-class `SoftwarePackage` is to cover the "logistics" aspect of the software installation procedure.⌋ *(RS_MANI_00035)*

Class	**SoftwarePackage**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to formalize the content of a software package. **Tags:** atp.Status=draft atp.recommendedPackage=SoftwarePackages			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
actionType	SoftwarePackageActionTypeEnum	1	attr	This attribute defines the action to be taken in the step of processing the enclosing SoftwarePackage.
compressedSoftwarePackageSize	PositiveInteger	1	attr	This size represents the size of the compressed SoftwarePackage.
deltaPackageApplicableVersion	StrongRevisionLabelString	0..1	attr	This attribute identifies the version of the included SoftwareCluster for which the enclosing SoftwarePackage can be used as a delta update
maximumSupportedUcmVersion	RevisionLabelString	1	attr	This attribute identifies the maximum supported version of the UCM for this SoftwarePackage.
minimumSupportedUcmVersion	RevisionLabelString	1	attr	This attribute identifies the minimum supported version of the UCM for this SoftwarePackage.
packagerId	PositiveInteger	1	attr	This attribute identifies Id of the organization that provides the packager generating the SoftwarePackage.
packagerSignature	CryptoServiceCertificate	1	ref	This reference identifies the certificate that represents the packager's signature. **Tags:**atp.Status=draft
postVerificationReboot	Boolean	1	attr	Reboot the platform after the verification of the activated software.
preActivate (ordered)	ModeDeclaration	*	iref	The referenced function group states shall be established for the switch between the already installed and the activated software. **Tags:**atp.Status=draft **InstanceRef implemented by:**FunctionGroupStateInFunctionGroupSetInstanceRef
preActivationReboot	Boolean	1	attr	Reboot the platform before the switch to the activated software.
softwareCluster	SoftwareCluster	1	ref	This reference identifies the SoftwareCluster that belongs to the SoftwarePackage. The nature of this relation is actually more like an aggregation than a reference. But the relation is still modelled as a reference because two ARElements cannot aggregate each other. **Tags:**atp.Status=draft
uncompressedSoftwareClusterSize	PositiveInteger	1	attr	This attribute gives an indication about the storage that has to be available on the target.

▽

△

Class	**SoftwarePackage**			
verify (ordered)	ModeDeclaration	*	iref	The referenced function group states shall be established for the verification of the activated software. **Tags:**atp.Status=draft **InstanceRef implemented by:**FunctionGroupStateIn FunctionGroupSetInstanceRef

Table 14.14: SoftwarePackage

Enumeration	**SoftwarePackageActivationActionEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This enumeration provides a choice of possible actions to be executed on installing a Software Package to a target Machine. **Tags:**atp.Status=draft
Literal	*Description*
reboot	Reboot the whole Machine. **Tags:**atp.EnumerationLiteralIndex=0
restartApplication	Restart the application software on the target Machine. **Tags:**atp.EnumerationLiteralIndex=1
waitForReboot	The installation has no immediate consequences in terms of other software on the target. **Tags:**atp.EnumerationLiteralIndex=2

Table 14.15: SoftwarePackageActivationActionEnum

[constr_1690]{DRAFT} **`SoftwareCluster` shall only be referenced by a single `SoftwarePackage`.** ⌈Each `SoftwareCluster` shall only be referenced by a single `SoftwarePackage`.⌋*()*

In other words, AUTOSAR factually assumes a 1:1 relation between `SoftwarePackage` and `SoftwareCluster`. Such a relation would otherwise typically be modeled by means of an aggregation with the multiplicity 1.

However, a `SoftwareCluster` is derived from base class `PackageableElement` which is only aggregated by `ARPackage`. Subclasses of `PackageableElement` – by convention – shall not be aggregated by any other meta-class.

[TPS_MANI_01222]{DRAFT} **Cryptographic signature of `SoftwarePackage`** ⌈A `SoftwarePackage` also needs to be signed cryptographically. For this purpose, meta-class `CryptoServiceCertificate` is referenced in the role `packagerSignature`.⌋*(RS_MANI_00035)*

[TPS_MANI_01223]{DRAFT} **Semantics of attribute `SoftwarePackage.packagerId`** ⌈Attribute `SoftwarePackage.packagerId` contains the value of the AUTOSAR vendor Id of the organization that created software tool that created the `SoftwarePackage`.⌋*(RS_MANI_00035)*

For clarification, a UCM can only accept packages that are generated by a packaging tool developed by the same organization that also developed the UCM itself. The vendor of the `SoftwareCluster` contained in the `SoftwarePackage` can obviously be different.

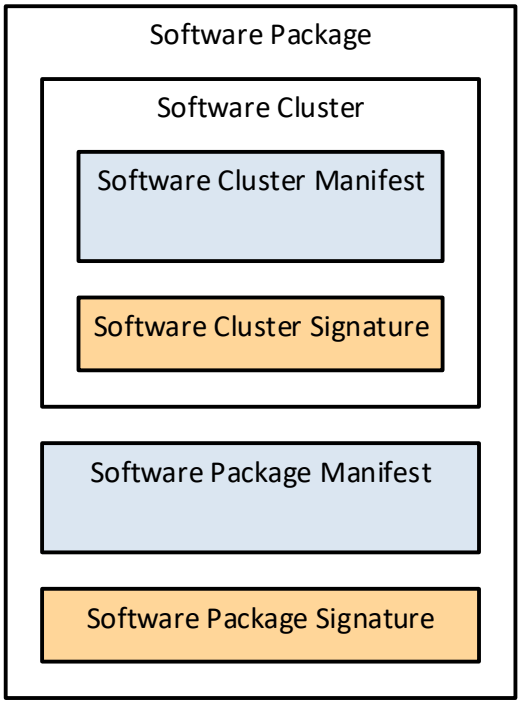


Figure 14.8: Conceptual relation of `SoftwarePackage` and `SoftwareCluster`

[TPS_MANI_01225]{DRAFT} **Actions taken during installation of a `SoftwarePackage`** ⌈It is necessary to define the concrete activity that shall be taken to handle the `SoftwarePackage` on the target machine. Possible actions are:

- Do a clean installation of a `SoftwareCluster`.
- Update a previously installed `SoftwareCluster`.
- Remove a `SoftwareCluster`

These options are formalized by means of meta-class `SoftwarePackageActionTypeEnum` and attribute `SoftwarePackage.actionType`.⌋*(RS_MANI_00035)*

Enumeration	**SoftwarePackageActionTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This enumeration provides a choice of possible actions for the handling of a software package. **Tags:**atp.Status=draft
Literal	***Description***
install	**Tags:**atp.EnumerationLiteralIndex=1
remove	**Tags:**atp.EnumerationLiteralIndex=2
update	**Tags:**atp.EnumerationLiteralIndex=0

Table 14.16: SoftwarePackageActionTypeEnum

The activation of a SoftwarePackage has an impact on the software that's already running on the target platform. This impact typically consists on the fulfillment of pre-requisites for the activation.

For example, if the intention is to update existing software then the existing software has to be steered into a neutral state before the activation can begin.

On the *AUTOSAR adaptive platform*, this translates to setting specific values for the function group states of specific function groups.

[TPS_MANI_01288]{DRAFT} **Impact of the SoftwarePackage on the value of function group states on the target platform** ⌈Typically, a SoftwarePackage needs to contain the information about which function group states are to be assumed as a prerequisite for the activation of the software and which function groups need to be assumed for the verification of the installed software on the target platform as a prerequisite for switching the software into a regular operation state.

Consequently, two references in the role SoftwarePackage.preActivate and SoftwarePackage.verify exist.⌋*()*

[TPS_MANI_01289]{DRAFT} **Order of function group states is relevant** ⌈The references SoftwarePackage.preActivate and SoftwarePackage.verify define the states of function groups for the pre-activation and the verification phase of the software activation.

However, the collection of function group states cannot be applied in arbitrary order. A given value of one function group state might only be achievable if another function group state has been reached previously. Therefore, the references preActivate and verify are modeled as ordered collections.⌋*()*

Additional reboots of the platform can be configured by means of attributes preActivationReboot and postVerificationReboot.

14.4 Vehicle Package

14.4.1 Overview

The ability to handle SoftwarePackages is the prerequisite for an important further step: the execution of an **update campaign that applies for the whole vehicle**. The basis for the update campaign is the definition of meta-class VehiclePackage.

[TPS_MANI_01290]{DRAFT} **VehiclePackage names affected UCMs** ⌈Meta-class VehiclePackage has the ability to describe the set of UCMs that are affected by the update campaign by means of aggregating meta-class UcmDescription in the role ucm.⌋*()*

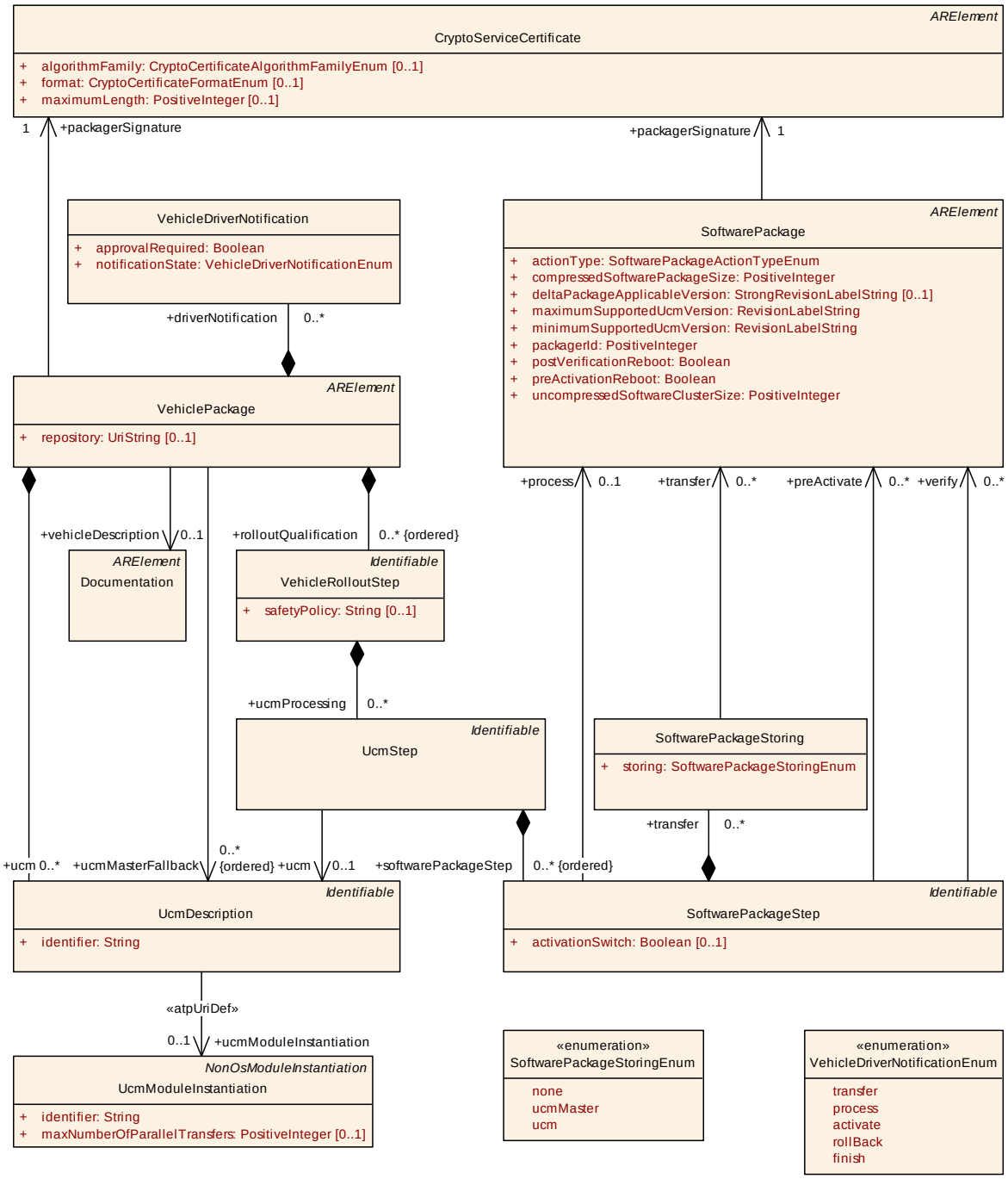


Figure 14.9: Modeling of `VehiclePackage`

Class	**VehiclePackage**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This meta-class represents the ability to define a vehicle package for executing an update campaign. **Tags:** atp.Status=draft atp.recommendedPackage=VehiclePackages

△

Class	VehiclePackage			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, Packageable Element, Referrable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
driver Notification	VehicleDriver Notification	*	aggr	This aggregation provides the ability to configure the necessary driver notifications. **Tags:**atp.Status=draft
packager Signature	CryptoService Certificate	1	ref	This reference identifies the certificate that represents the packager's signature. **Tags:**atp.Status=draft
repository	UriString	0..1	attr	This attribute identifies the repository where the Vehicle Package is stored.
rollout Qualification (ordered)	VehicleRolloutStep	*	aggr	This represents the rollout qualification. **Tags:**atp.Status=draft
ucm	UcmDescription	*	aggr	This aggregation represents the UcmDescriptions to be considered in the context of the VehiclePackage. **Tags:**atp.Status=draft
ucmMaster Fallback (ordered)	UcmDescription	*	ref	This reference lists the fallback order of Ucms that can take over the master role if the master goes down. **Tags:**atp.Status=draft
vehicle Description	Documentation	0..1	ref	This reference identifies the vehicle description. **Tags:**atp.Status=draft

Table 14.17: VehiclePackage

Class	UcmDescription			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define an identifier for a given UCM. **Tags:**atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
identifier	String	1	attr	This attribute represents the unique identification of the UcmIdentifier.
ucmModule Instantiation	UcmModuleInstantiation	0..1	ref	This reference identifies the applicable UcmModule Instantiation. **Stereotypes:** atpUriDef **Tags:**atp.Status=draft

Table 14.18: UcmDescription

[TPS_MANI_01291]{DRAFT} **Identification of an actual UCM in the context of an update campaign** ⌈It is necessary to unambiguously identify the individual UCMs that are affected in the update campaign. For this purpose, meta-class `UcmDescription` defines attribute `identifier`.

By means of the reference to `UcmModuleInstantiation` in the role `ucmModule-Instantiation` it is in addition possible to identify the actual UCMs (represented by a `UcmModuleInstantiation`) that are relevant for the update campaign.

In order to be able to resolve the reference it is necessary to have access to the manifest model of the target Machine.⌋()

[constr_1731]{DRAFT} **Value of UcmDescription.identifier in the scope of a VehiclePackage** ⌈Within the scope of any given VehiclePackage, no two UcmDescriptions shall define the same value of attribute identifier.⌋()

[TPS_MANI_01292]{DRAFT} **Definition of fallback-order for UCM master** ⌈The update campaign is executed under the management of one UCM that acts as a "master UCM".

If this UCM goes down for some reason, VehiclePackage has the ability to define an **ordered** list of other candidates for becoming the "master UCM" by means of the reference to meta-class UcmDescription in the role ucmMasterFallback.⌋()

[TPS_MANI_01294]{DRAFT} **Update campaign depends on driver's acceptance** ⌈For obvious reasons, it is not possible to arbitrarily trigger the execution of an update campaign at any time. It is the prerogative of the vehicle driver to decide about the amount and consequence of the UCM activities with respect to an update campaign.

For this purpose VehiclePackage aggregates meta-class VehicleDriverNotification in the role driverNotification.⌋()

Class	**VehicleDriverNotification**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class provides the ability to configure a notification of the vehicle driver with respect to the update of vehicle software. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
approval Required	Boolean	1	attr	This attribute controls whether approval is required for the driver notification.
notificationState	VehicleDriver NotificationEnum	1	attr	This attribute is used to configure the notification state.

Table 14.19: VehicleDriverNotification

Enumeration	**VehicleDriverNotificationEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This meta-class provides available options for vehicle driver notification. **Tags:**atp.Status=draft
Literal	***Description***
activate	Software package shall be activated. **Tags:**atp.EnumerationLiteralIndex=2
finish	Finish notification **Tags:**atp.EnumerationLiteralIndex=4
process	Processing of software package shall be executed **Tags:**atp.EnumerationLiteralIndex=1

▽

△

Enumeration	**VehicleDriverNotificationEnum**
rollBack	Software package shall be rolled back. **Tags:**atp.EnumerationLiteralIndex=3
transfer	Software shall be transferred to the vehicle. **Tags:**atp.EnumerationLiteralIndex=0

Table 14.20: VehicleDriverNotificationEnum

14.4.2 VehicleRolloutStep

[TPS_MANI_01295]{DRAFT} **Semantics of `VehicleRolloutStep`** ⌈The purpose of an update campaign is to roll out the installation or update of `SoftwarePackage`s in the context of given UCMs. Each `VehicleRolloutStep` may apply to several UCMs at the same time.⌋*()*

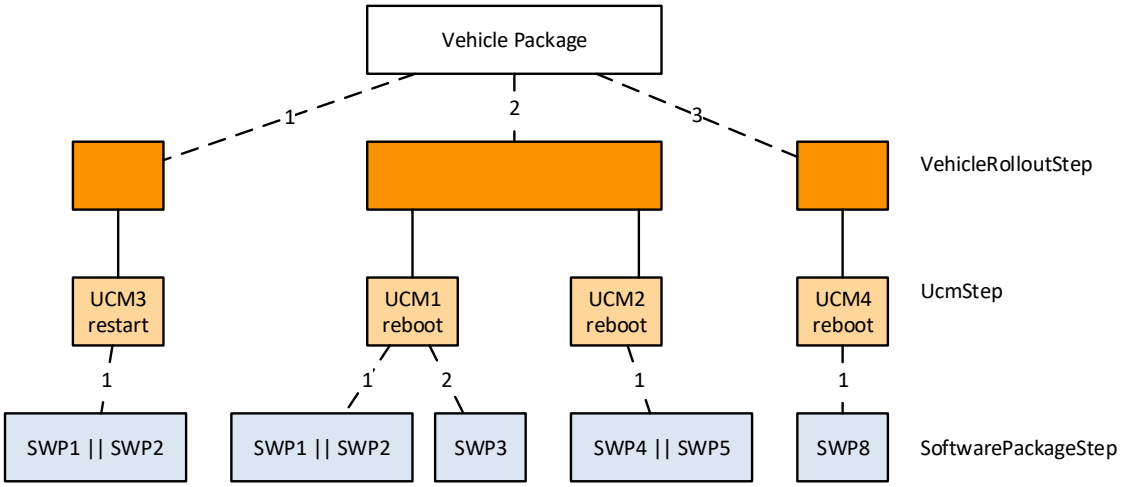


Figure 14.10: Conceptual view on an update campaign

Figure 14.10 takes a conceptual view on the structure of an update campaign and associates the relevant aspects of this view to meta-classes explained in this chapter. Associations that are labeled by a number indicate that an ordering is implied with the respective step.

For example, the execution of the update campaign happens in dedicated steps formalized as `VehicleRolloutStep`, as explained in [TPS_MANI_01295]. Each of the three steps sketched in the picture would be modeled as an individual `VehicleRolloutStep`.

The rollout action is formalized by `UcmStep.softwarePackageStep`. In other words, it is possible to specify a different `softwarePackageStep` for each individual UCM.

The individual VehicleRolloutSteps are executed in the order in which they are aggregated at the enclosing VehiclePackage. This aspect is in more detail explained by [TPS_MANI_01296].

Class	**VehicleRolloutStep**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define a rollout-condition for a vehicle update campaign. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
safetyPolicy	String	0..1	attr	This attribute contains the applicable safety policy for the enclosing SoftwarePackageStep.
ucmProcessing	UcmStep	*	aggr	This aggregation collects the UcmProcessingSteps that make up the rollout step. **Tags:**atp.Status=draft

Table 14.21: VehicleRolloutStep

[TPS_MANI_01296]{DRAFT} **Ordered execution of rollout steps in an update campaign** ⌈The individual VehicleRolloutSteps defined in the context of a given VehiclePackage are executed in the defined order and therefore the aggregation of VehicleRolloutStep at VehiclePackage is ordered.⌋*()*

14.4.3 UcmStep

[TPS_MANI_01297]{DRAFT} **Semantics of meta-class UcmStep** ⌈Each VehicleRolloutStep consists of a number of UcmSteps (aggregated by VehicleRolloutStep in the role ucmProcessing). Each UcmStep refers to a specific UCM (represented by UcmDescription) in the role ucm.

The campaign is executed in steps formalized as UcmStep that each are associated with a specific activation action (restart a SoftwareCluster or reboot the entire Machine) formalized by means of softwarePackageStep.⌋*()*

[TPS_MANI_01298]{DRAFT} **No ordering of VehicleRolloutStep.ucmProcessing** ⌈Each UcmStep defined in the context of an enclosing VehicleRolloutStep can be handled without the consideration of a dedicated order. Therefore, the aggregation VehicleRolloutStep.ucmProcessing is not labeled as ordered.⌋*()*

Class	**UcmStep**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This meta-class represents the ability to define a rollout-condition for a vehicle update campaign. **Tags:**atp.Status=draft
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	UcmStep			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
software PackageStep (ordered)	SoftwarePackageStep	*	aggr	This aggregation represents the sequence of activities to be carried out in the context of the respective UCM. **Tags:**atp.Status=draft
ucm	UcmDescription	0..1	ref	This reference identifies the UCM for which the rollout step applies. **Tags:**atp.Status=draft

Table 14.22: UcmStep

14.4.4 SoftwarePackageStep

[TPS_MANI_01299]{DRAFT} **Aggregation of SoftwarePackageSteps at UcmStep** ⌈Each UcmStep consists of an ordered collection of SoftwarePackageSteps. This means that the order in which SoftwarePackages are handled in the scope of one UcmStep is significant.⌋*()*

Class	SoftwarePackageStep			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the configuration of an activation step in the context of software package activation. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
activationSwitch	Boolean	0..1	attr	If set to True, the enclosing SoftarePackageStep represents the actual activation of all affected Software Packages.
preActivate	SoftwarePackage	*	ref	This reference identifies the SoftwarePackage to be pre-activated in the enclosing SoftwarePackageStep. **Tags:**atp.Status=draft
process	SoftwarePackage	0..1	ref	This reference identifies the SoftwarePackage to be processed in the enclosing SoftwarePackageStep. **Tags:**atp.Status=draft
transfer	SoftwarePackage Storing	*	aggr	This aggregation clarifies the storing of the Software Package. **Tags:**atp.Status=draft
verify	SoftwarePackage	*	ref	This reference identifies the SoftwarePackage to be verified in the enclosing SoftwarePackageStep. **Tags:**atp.Status=draft

Table 14.23: SoftwarePackageStep

[TPS_MANI_01300]{DRAFT} **Semantics of reference SoftwarePackageStep.transfer.transfer** ⌈The reference SoftwarePackageStep.transfer.transfer identifies the SoftwarePackages that are supposed to be transferred in the context of the enclosing SoftwarePackageStep.

It is positively supported that SoftwarePackages are transferred in parallel and therefore the multiplicity of the reference in the role transfer has been set to 0..*.⌋()

[TPS_MANI_01301]{DRAFT} **Semantics of aggregation SoftwarePackageStep.transfer** ⌈By means of the aggregation of SoftwarePackageStoring it is possible to specify for each individual SoftwarePackage to specify whether and where the SoftwarePackage is stored in the vehicle.

This information is specifically provided by attribute SoftwarePackageStoring.storing of type SoftwarePackageStoringEnum.⌋()

Class	**SoftwarePackageStoring**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class provides the ability to specify whether and where the referenced SoftwarePackage is stored. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
storing	SoftwarePackage StoringEnum	1	attr	This attribute clarifies whether and where the referenced SoftwarePackage is stored.
transfer	SoftwarePackage	*	ref	This reference identifies the SoftwarePackage(s) to be transferred in the enclosing SoftwarePackageStep. **Tags:**atp.Status=draft

Table 14.24: SoftwarePackageStoring

Enumeration	**SoftwarePackageStoringEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	**Tags:**atp.Status=draft
Literal	***Description***
none	No storing in vehicle. **Tags:**atp.EnumerationLiteralIndex=0
ucm	Storing in UCM (subordinate). **Tags:**atp.EnumerationLiteralIndex=2
ucmMaster	Storing in Ucm Master. **Tags:**atp.EnumerationLiteralIndex=1

Table 14.25: SoftwarePackageStoringEnum

[TPS_MANI_01302]{DRAFT} **Semantics of reference SoftwarePackageStep.process** ⌈The reference SoftwarePackageStep.process identifies the SoftwarePackage that is supposed to be processed in the context of the enclosing SoftwarePackageStep.

The processing of SoftwarePackages happens strictly one after the other and therefore the reference process can only have the multiplicity 0..1. The strict order of processing is guaranteed by the aggregation of the SoftwarePackageStep at UcmStep.⌋()

[TPS_MANI_01303]{DRAFT} **Semantics of reference SoftwarePackageStep.preActivate** ⌈The reference SoftwarePackageStep.preActivate identifies the SoftwarePackages that are supposed to be pre-activated in the context of the enclosing SoftwarePackageStep.

The order of the pre-activation in the context of one SoftwarePackageStep is not defined and may be arbitrary.

If this is not acceptable then the pre-activation of a set of SoftwarePackages shall be modelled in the context of different SoftwarePackageSteps that can be arranged in the intended order.⌋*()*

[TPS_MANI_01304]{DRAFT} **Semantics of reference SoftwarePackageStep.verify** ⌈The reference SoftwarePackageStep.verify identifies the SoftwarePackages that are supposed to be verified in the context of the enclosing SoftwarePackageStep.

The order of the verification in the context of one SoftwarePackageStep is not defined and may be arbitrary.

If this is not acceptable then the verification of a set of SoftwarePackages shall be modelled in the context of different SoftwarePackageSteps that can be arranged in the intended order.⌋*()*

[TPS_MANI_01305]{DRAFT} **Semantics of attribute SoftwarePackageStep.activationSwitch** ⌈The usage of attribute activationSwitch in a given SoftwarePackageStep indicates that the enclosing SoftwarePackageStep represents the *activation switch step* in the activation workflow.⌋*()*

[constr_1732]{DRAFT} **Existence of attribute activationSwitch set to True in the context of the enclosing UcmStep** ⌈Within the context of any given UcmStep, only a single SoftwarePackageStep shall exist that sets the value of attribute activationSwitch to the value True.⌋*()*

[TPS_MANI_01306]{DRAFT} **Simultaneous existence of attributes SoftwarePackageStep.transfer and SoftwarePackageStep.process** ⌈It is possible that the references SoftwarePackageStep.transfer and SoftwarePackageStep.process simultaneously exist to the identical SoftwarePackage in the context of the same SoftwarePackageStep.

The semantics of such a configuration is that the SoftwarePackage that is referenced by the two roles owned by the same SoftwarePackageStep is "streamed", i.e. transferred and processed in one step represented by the SoftwarePackageStep.⌋*()*

[constr_1733]{DRAFT} **Simultaneous existence of SoftwarePackageStep.preActivate and SoftwarePackageStep.verify** ⌈The references SoftwarePackageStep.preActivate and SoftwarePackageStep.verify shall not be defined in the context of the same enclosing SoftwarePackageStep⌋*()*

[constr_1734]{DRAFT} **Restriction for attribute SoftwarePackageStep.activationSwitch** ⌈If attribute SoftwarePackageStep.activationSwitch is set

to `True` then the enclosing `SoftwarePackageStep` shall not refer to a `SoftwarePackage` in any of the possible roles.⌋*()*

14.4.5 Examples for the Usage of SoftwarePackageStep

The semantics of the references

- `transfer`
- `process`
- `preActivate`
- `verify`

shall be explained along a set of examples.

14.4.5.1 Examples for the Usage of transfer and process

The first example (as depicted in Figure 14.11) assumes a scenario where three `SoftwarePackage`s shall be activated and the three `SoftwarePackage`s are transferred in parallel in the context of one `SoftwarePackageStep`.

After that, the three `SoftwarePackage`s are processed one by one in the context of three further `SoftwarePackageStep`s.

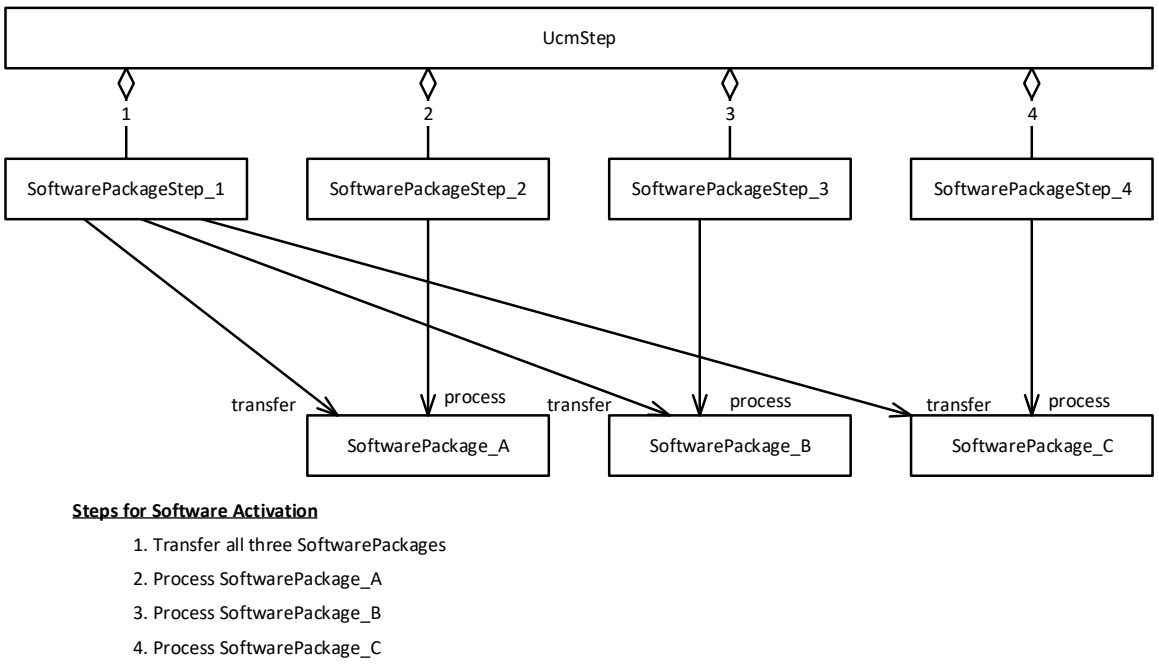


Figure 14.11: Example 1 of the configuration of `SoftwarePackageStep`

The ordering of the SoftwarePackageSteps in the context of the enclosing UcmStep is depicted explicitly by assigning numerical values to the aggregation as well as naming the respective SoftwarePackageStep using the same numerical values.

The second example introduces a scenario where the three SoftwarePackages are transferred in the first SoftwarePackageStep. One of the SoftwarePackages is additionally referenced in the role process and this indicates that the respective SoftwarePackage is streamed.

Because the first SoftwarePackage is streamed, only three[1] SoftwarePackageSteps are needed to model the example.

The rest of the SoftwarePackages are processed in a dedicated SoftwarePackageStep. The example is depicted in Figure 14.12.

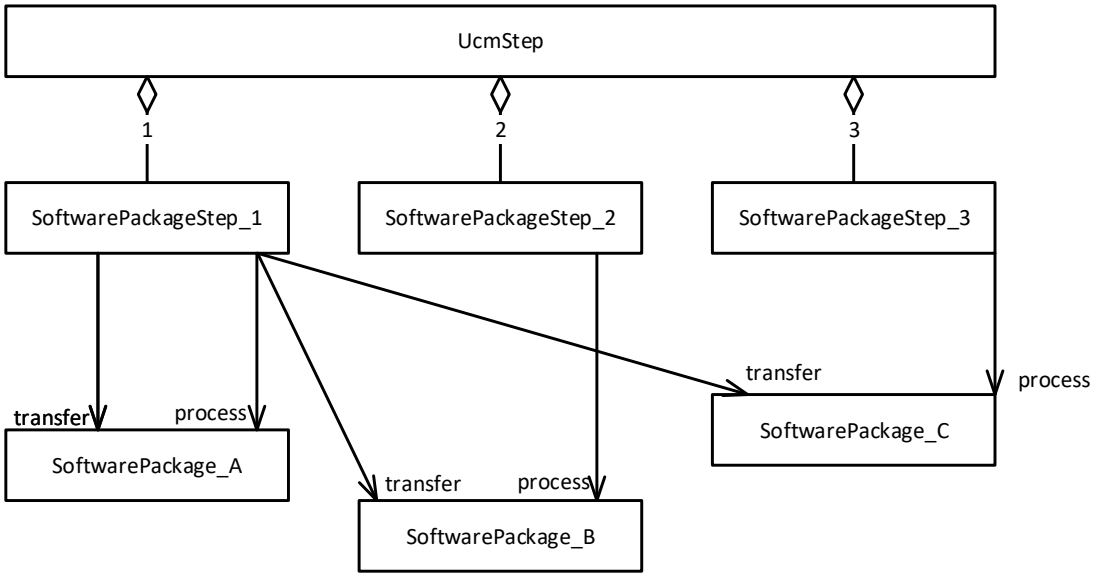


Figure 14.12: Example 2 of the configuration of SoftwarePackageStep

The third example, depicted in Figure 14.13, sketches a scenario where three SoftwarePackages are streamed one after the other.

Consequently, this scenario requires the existence of three SoftwarePackageSteps that each reference the same SoftwarePackage in the roles transfer and process.

[1] As opposed to four SoftwarePackageSteps required to model the first example.

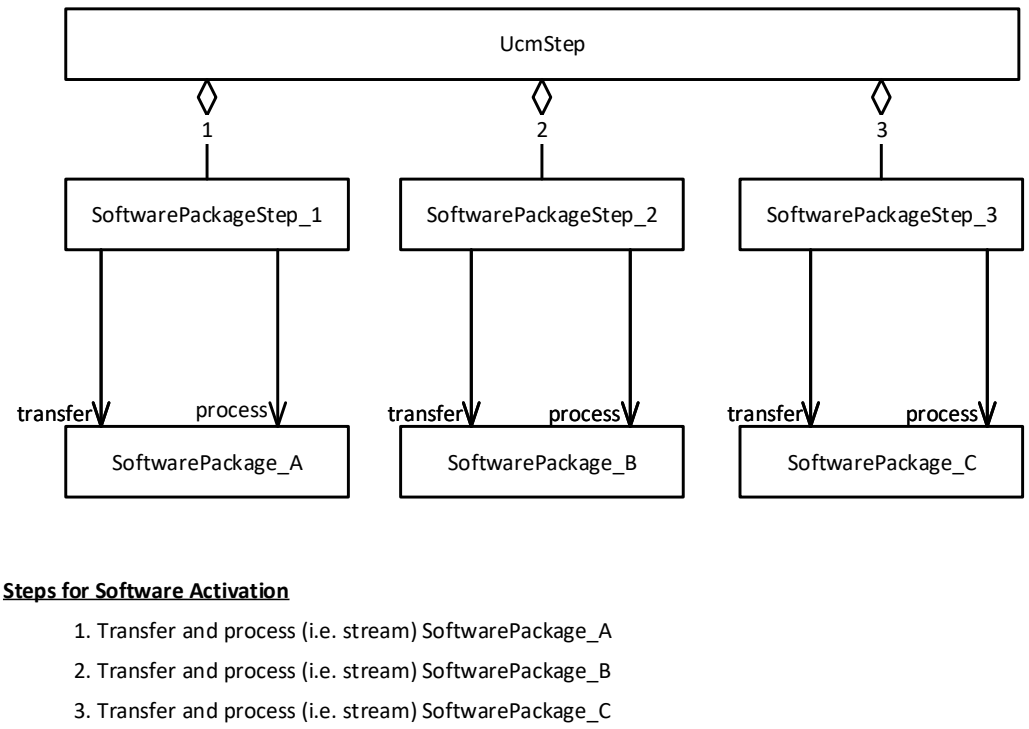


Figure 14.13: Example 3 of the configuration of `SoftwarePackageStep`

14.4.5.2 Examples for the Usage of pre-activate and verify

A more complex example of a software activation workflow is presented “flipbook-style” to emphasize the individual steps.

As depicted in Figure 14.14, step 1 of this scenario (formalized by the aggregation of a `SoftwarePackageStep` named `SoftwarePackageStep_1` at the enclosing `UcmStep`) in the activation workflow is the transfer of two `SoftwarePackage`s named `SoftwarePackage_1` and `SoftwarePackage_2`.

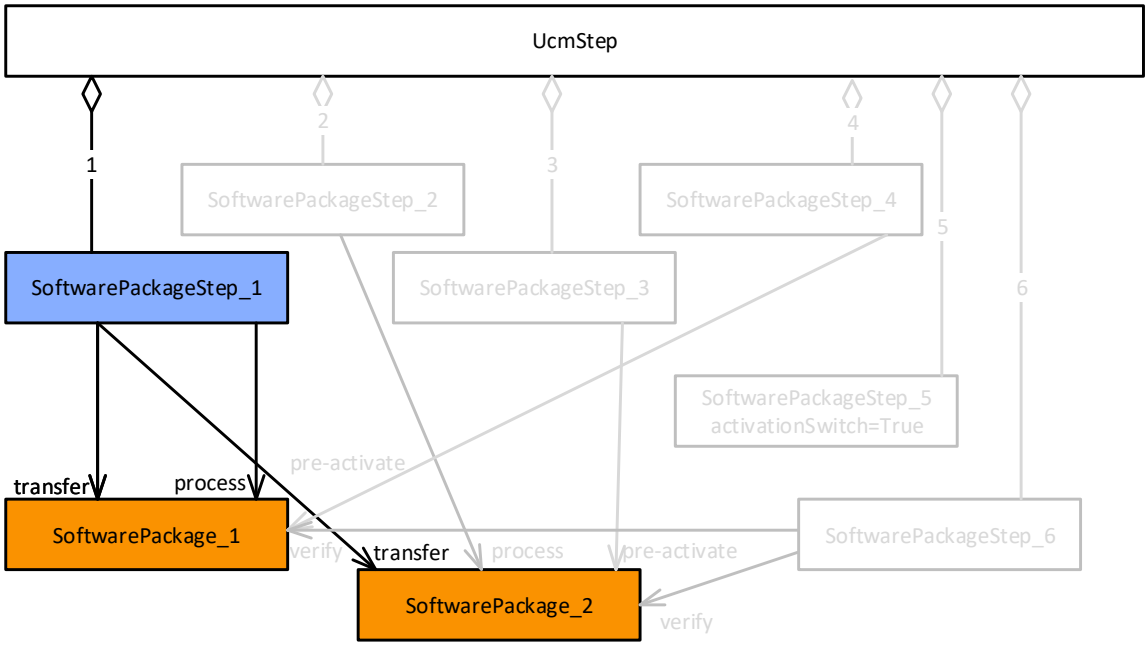


Figure 14.14: Step 1 in the activation workflow

In the same step, `SoftwarePackage_1` is also processed. The second step (see Figure 14.15) consists of the processing of `SoftwarePackage_2`.

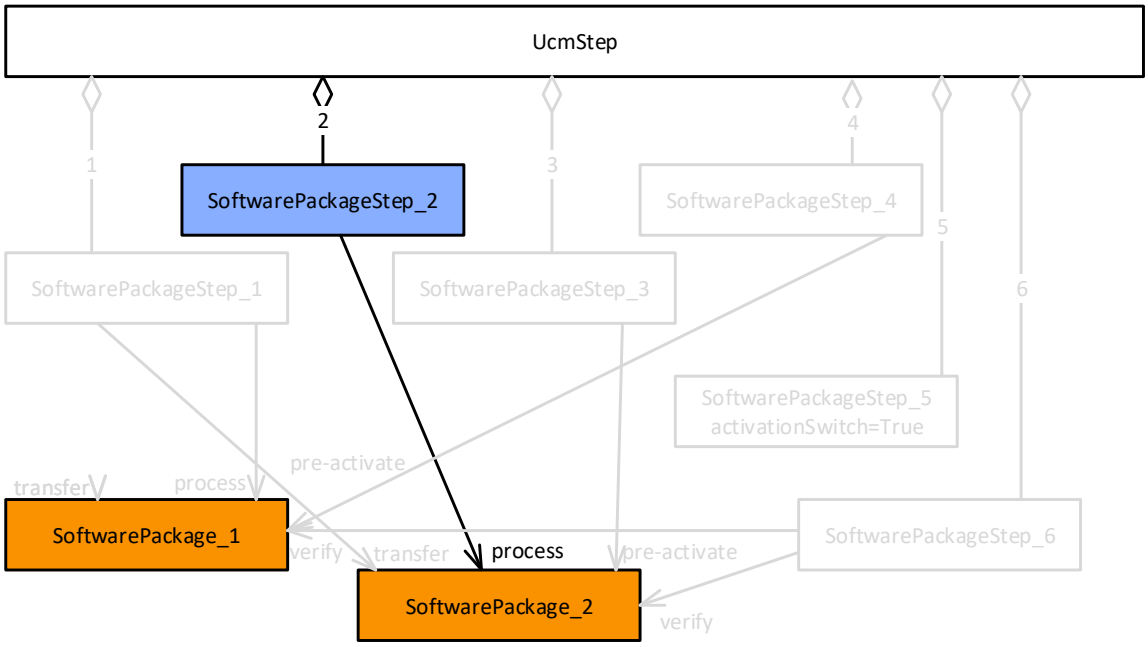


Figure 14.15: Step 2 in the activation workflow

In the third step (see Figure 14.16), `SoftwarePackage_2` gets pre-activated while in the fourth step (see Figure 14.17) the same applies for `SoftwarePackage_1`.

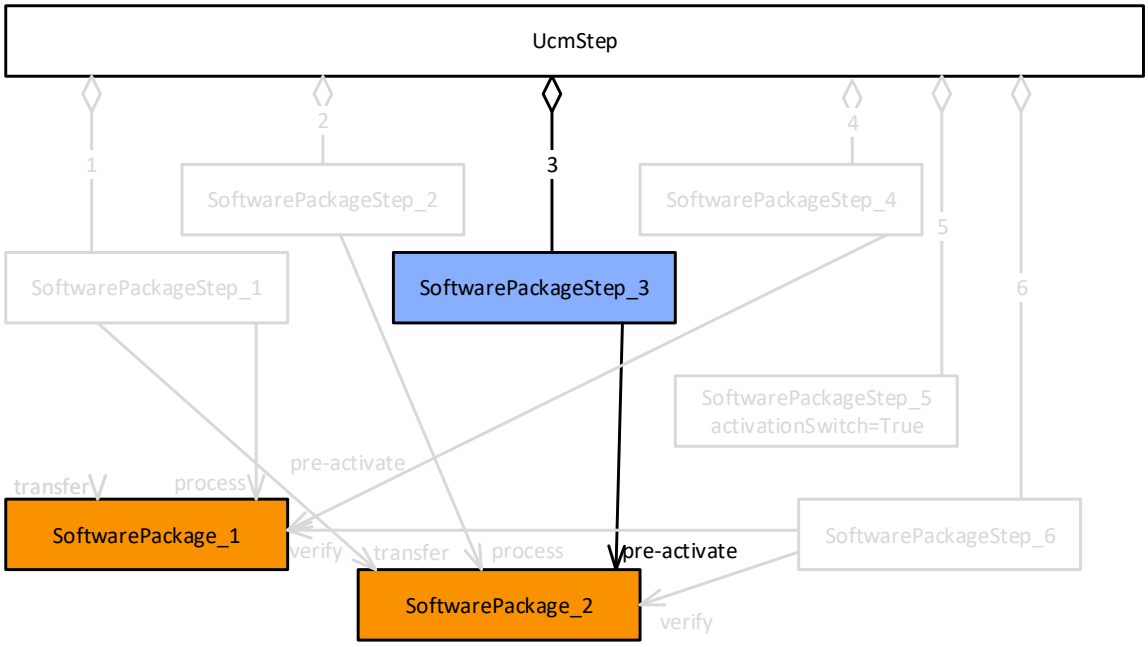


Figure 14.16: Step 3 in the activation workflow

Please note that the modeling of step 3 (see Figure 14.16) and 4 (see Figure 14.17)has been created to emphasize that the order in which `SoftwarePackage`s can be pre-activated is not necessarily identical to the order in which they have been processed. By the demonstrated modeling, it is possible to implement any order whatsoever.

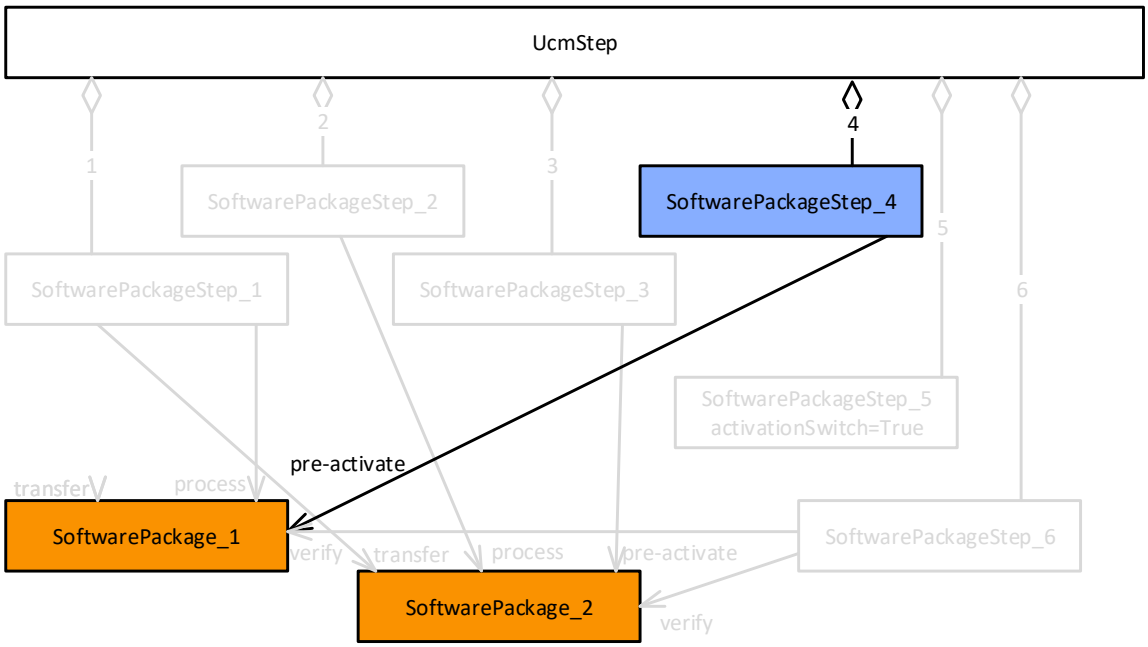


Figure 14.17: Step 4 in the activation workflow

The fifth step (see Figure 14.18) marks the actual activation of the two SoftwarePackages. For this purpose the attribute SoftwarePackageStep.activationSwitch is set to True. Consequently, this SoftwarePackageStep does not reference any of the SoftwarePackages in any of the supported roles.

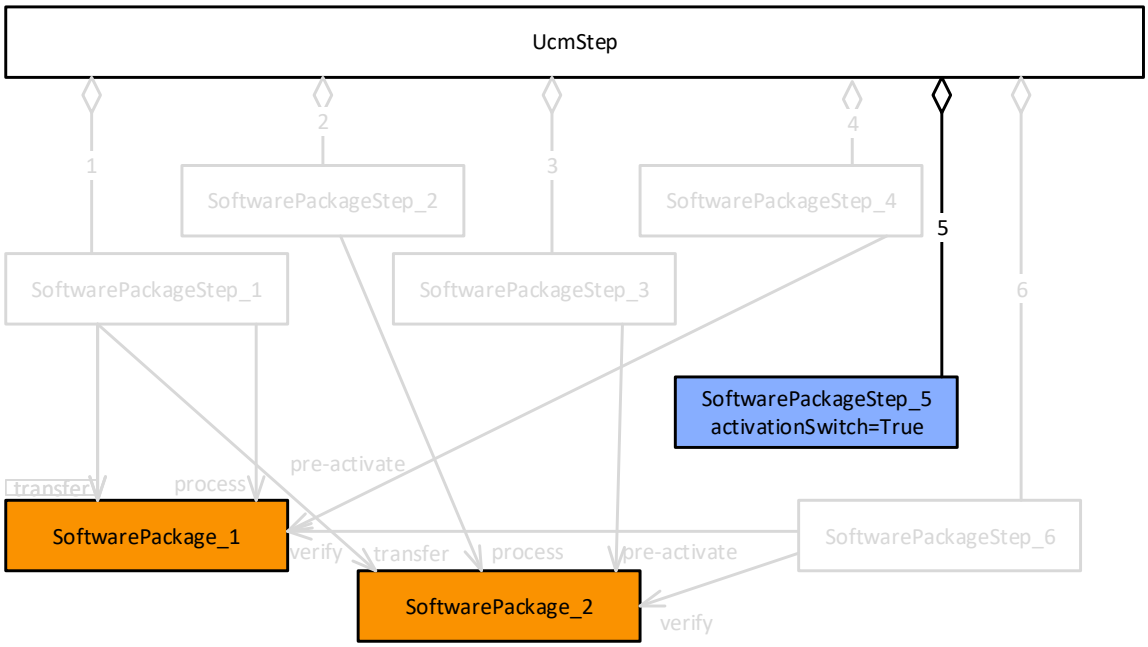


Figure 14.18: Step 5 in the activation workflow

In the sixth step (see Figure 14.19) finalizes the workflow by verifying the two SoftwarePackages named SoftwarePackage_1 and SoftwarePackage_2. The modeling of the verification in one step indicates that the actual order in which the SoftwarePackages is not relevant.

If, in contrast to the depicted scenario, the order were relevant then it would take the modeling of two different SoftwarePackageSteps to codify the intended ordering.

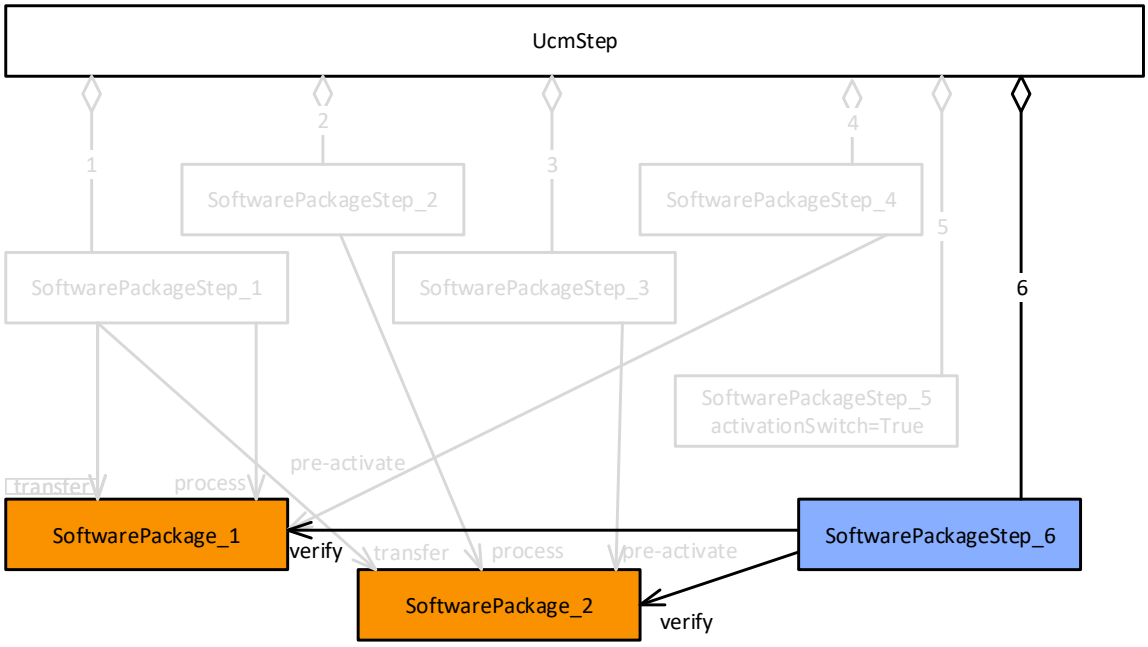


Figure 14.19: Step 6 in the activation workflow

A Examples

This chapter contains a collection of examples that reflect concepts described in different chapters of this document. The content of the chapter provides mere explanation and does not add anything to the model semantics.

A.1 Service Instance Deployment by Service Interface Mapping

The example in Figure A.2 sketches the modeling of a `ProvidedSomeipServiceInstance` in the presence of a `ServiceInterfaceMapping`, that references two `ServiceInterface`s in the role `sourceServiceInterface`.

For support, Figure A.1 contains an excerpt from the meta-model that contains the relevant meta-classes that have been instantiated to create the example sketched in Figure A.2.

Note further that the example depicted in Figure A.2 is not limited to the explanation of the actual `ServiceInterfaceMapping`.

As the main use case for this is the usage of `ServiceInterface`s for the definition of an “outside” communication binding the example also contains the modeling of such a binding, in this case to SOME/IP.

Please note that the modeling of the binding requires the existence of a `PortPrototype`, which in turn is aggregated by an `SwComponentType` (not depicted).

This approach still contains some degrees of freedom with respect to the role of the `SwComponentType` that aggregates the mentioned `PortPrototype`. This document does not go further in discussing the nature of such a configuration.

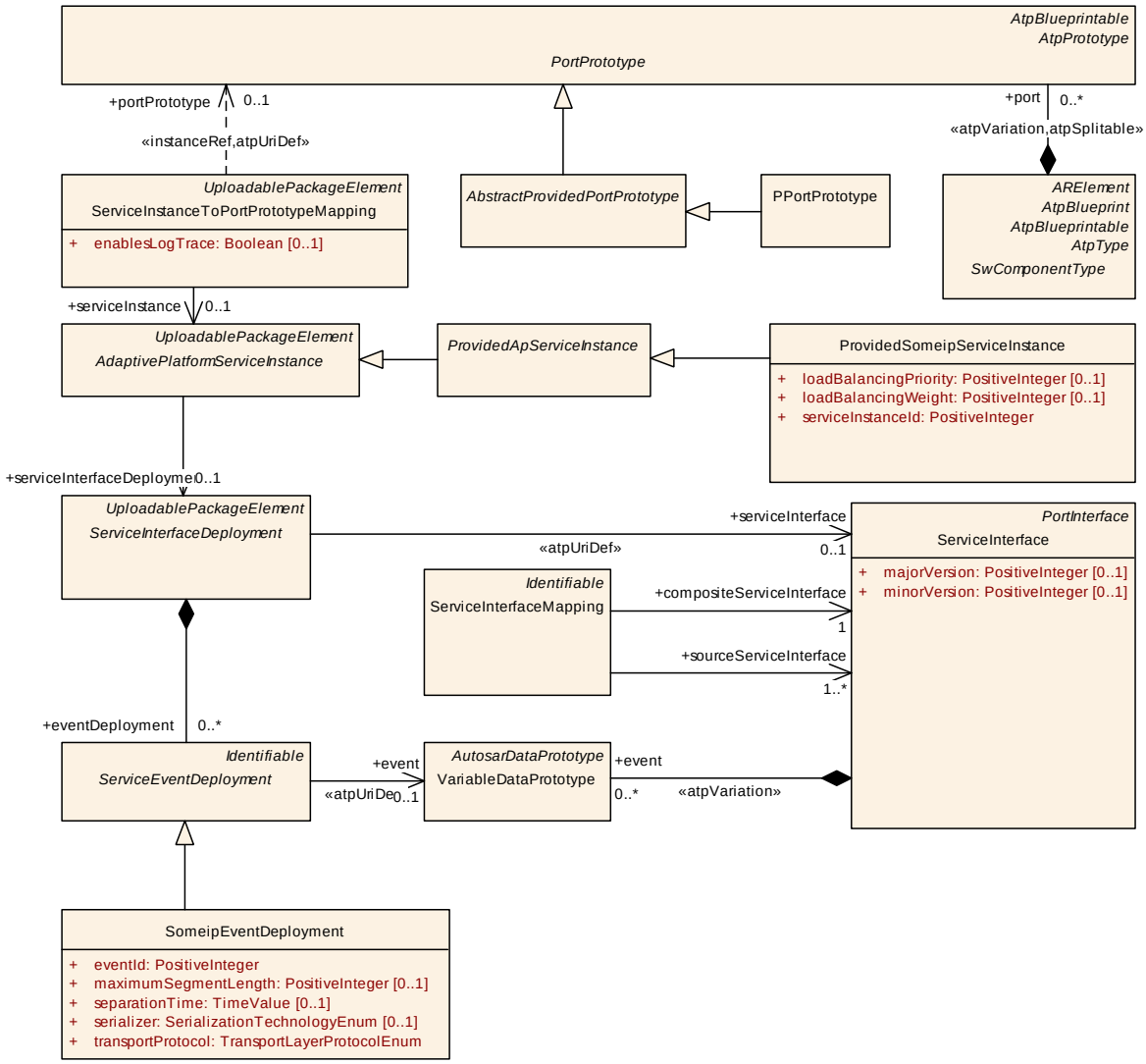


Figure A.1: Meta-model excerpt relevant for the example

For reasons of keeping the example as simple as possible, each of the `ServiceInterfaces` in the role `sourceServiceInterface` aggregate a single `event`.

The `ServiceInterface` referenced in the role `compositeServiceInterface` aggregates two `event` with `shortName`s that match the mentioned `event` of the source `ServiceInterfaces` (see [TPS_MANI_01022]).

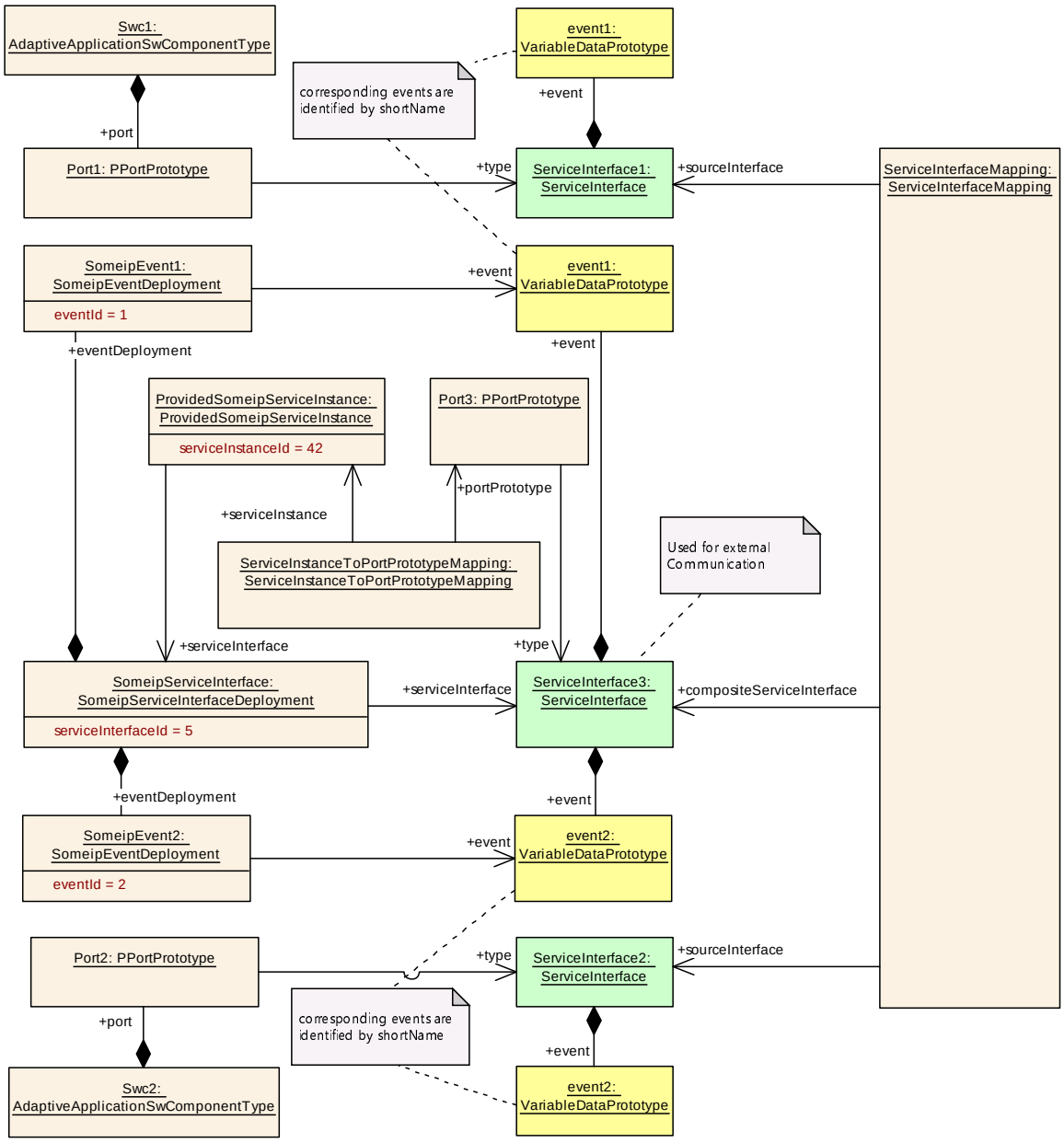


Figure A.2: Example for the deployments of a service in the presence of a `ServiceInterfaceMapping`

A.2 Service Instance Deployment by Service Interface Element Mapping

The example in Figure A.4 sketches the modeling of a `ProvidedSomeipServiceInstance` in the presence of a `ServiceInterfaceEventMapping`s. In principle, this example is very close to the example described in Figure A.2.

In contrast to the example sketched in Figure A.2, the example depicted in Figure A.4 uses a mapping to individual elements of a ServiceInterface instead of the entire ServiceInterface.

Please find the corresponding excerpt of relevant meta-classes for the utilization of ServiceInterfaceEventMapping sketched in Figure A.3.

Note further that the example depicted in Figure A.3 is not limited to the explanation of the actual ServiceInterfaceElementMapping.

As the main use case for this is the usage of ServiceInterfaces for the definition of an “outside” communication binding the example also contains the modeling of such a binding, in this case to SOME/IP.

Please note that the modeling of the binding requires the existence of a PortPrototype, which in turn is aggregated by an SwComponentType (not depicted).

This approach still contains some degrees of freedom with respect to the role of the SwComponentType that aggregates the mentioned PortPrototype. This document does not go further in discussing the nature of such a configuration.

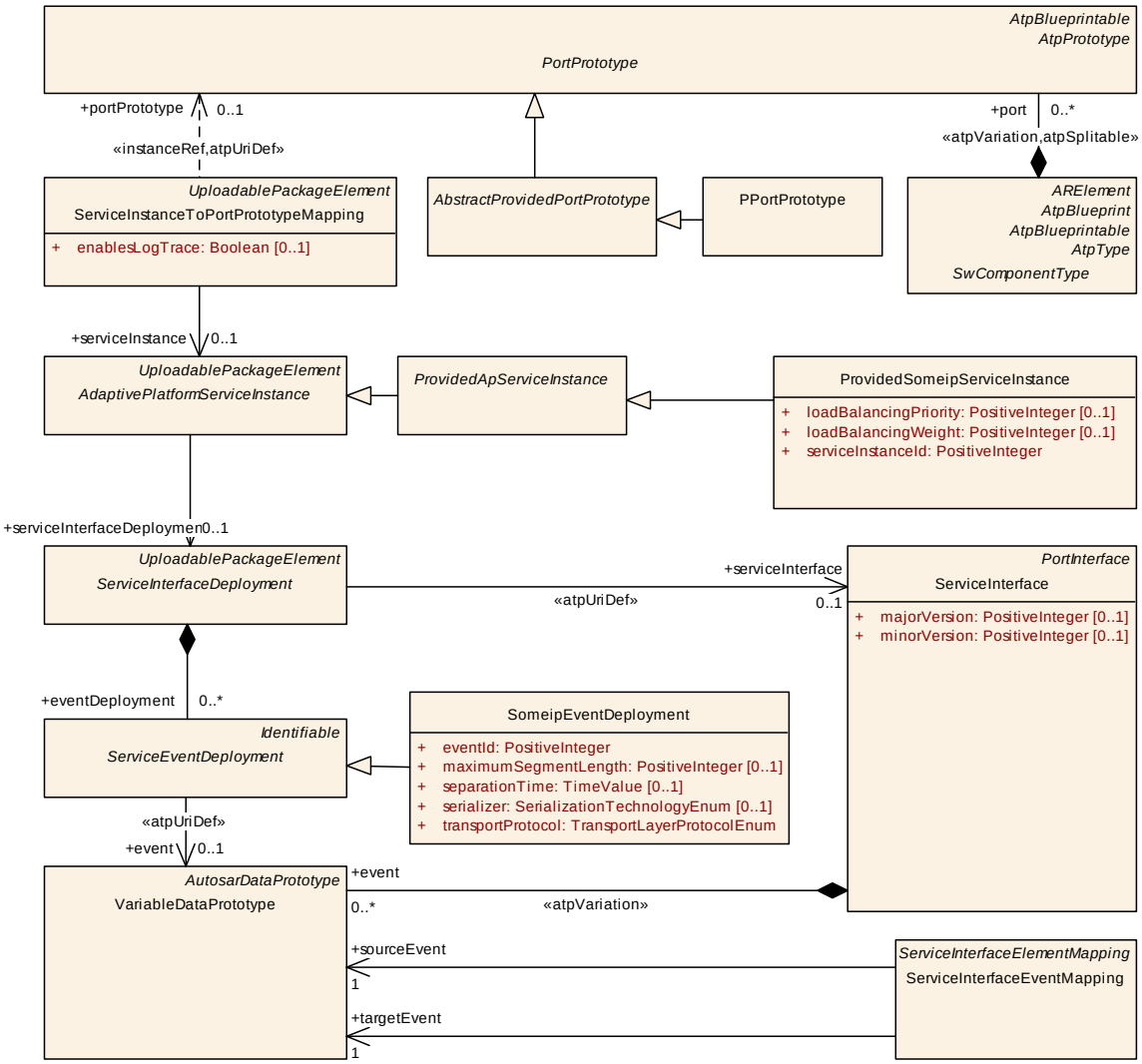


Figure A.3: Excerpt of the relevant meta-classes for the `ServiceInterfaceEventMapping` example

By mapping individual elements of `ServiceInterface`s, it is possible to map element with different `shortName`s to each other. In this example, the `event` with the `shortName` `event1` is mapped to another `event` with the `shortName` `eventLeft`.

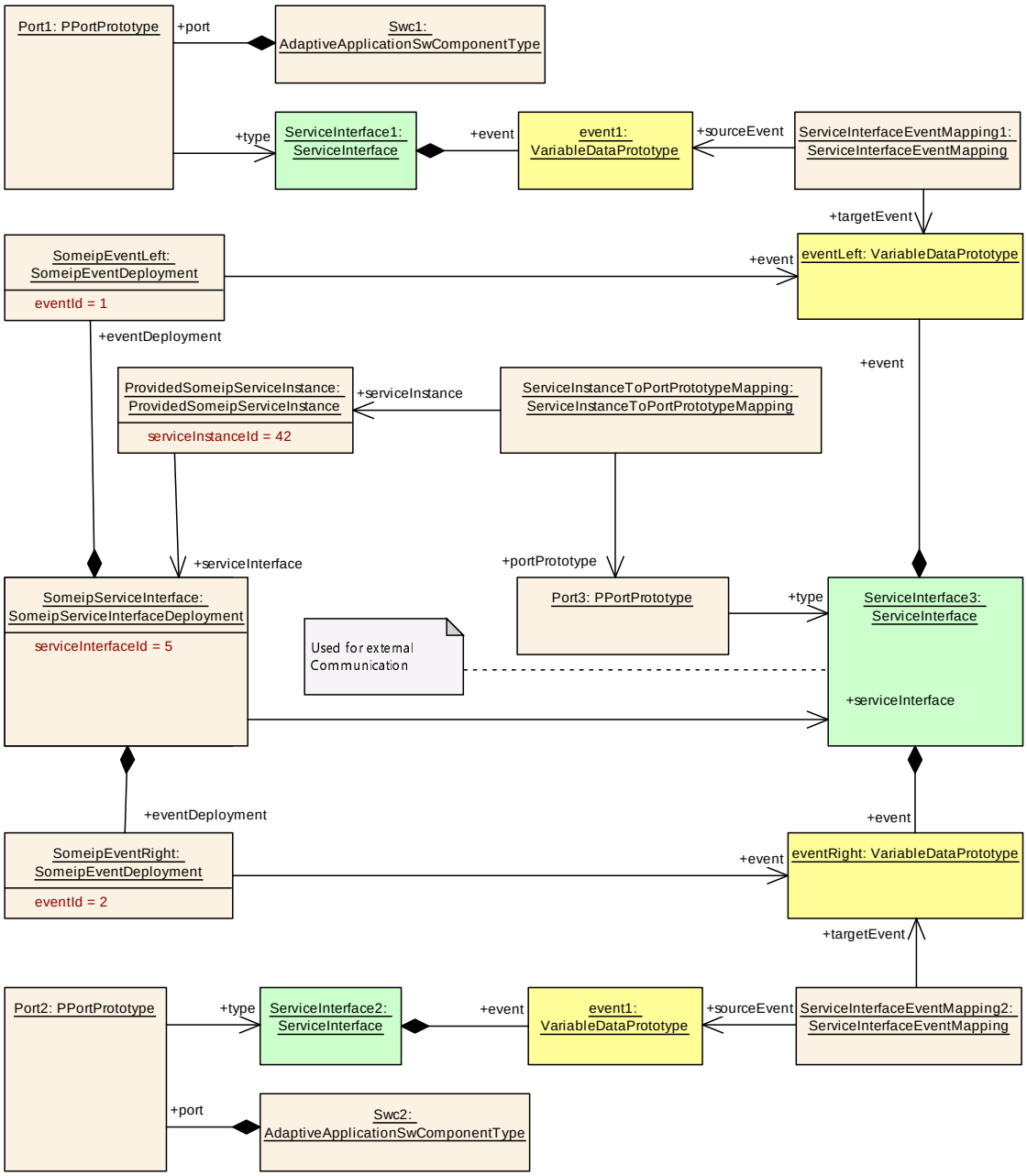


Figure A.4: Example for the deployment of a service in the presence of a `ServiceInterfaceEventMapping`

In Figure A.4, two different `ServiceInterface`s exist that each aggregate an `event` with the identical `shortName`. This scenario **requires** the existence of `ServiceInterfaceElementMapping`s.

As an extension to the scenario depicted in Figure A.4, Figure A.5 describes a model where the **same** `event` of a `ServiceInterface` is used in two different event deployments by means of two `ServiceInterfaceEventMapping`s that each refer to said `event` in the role `ServiceInterfaceEventMapping.sourceEvent`.

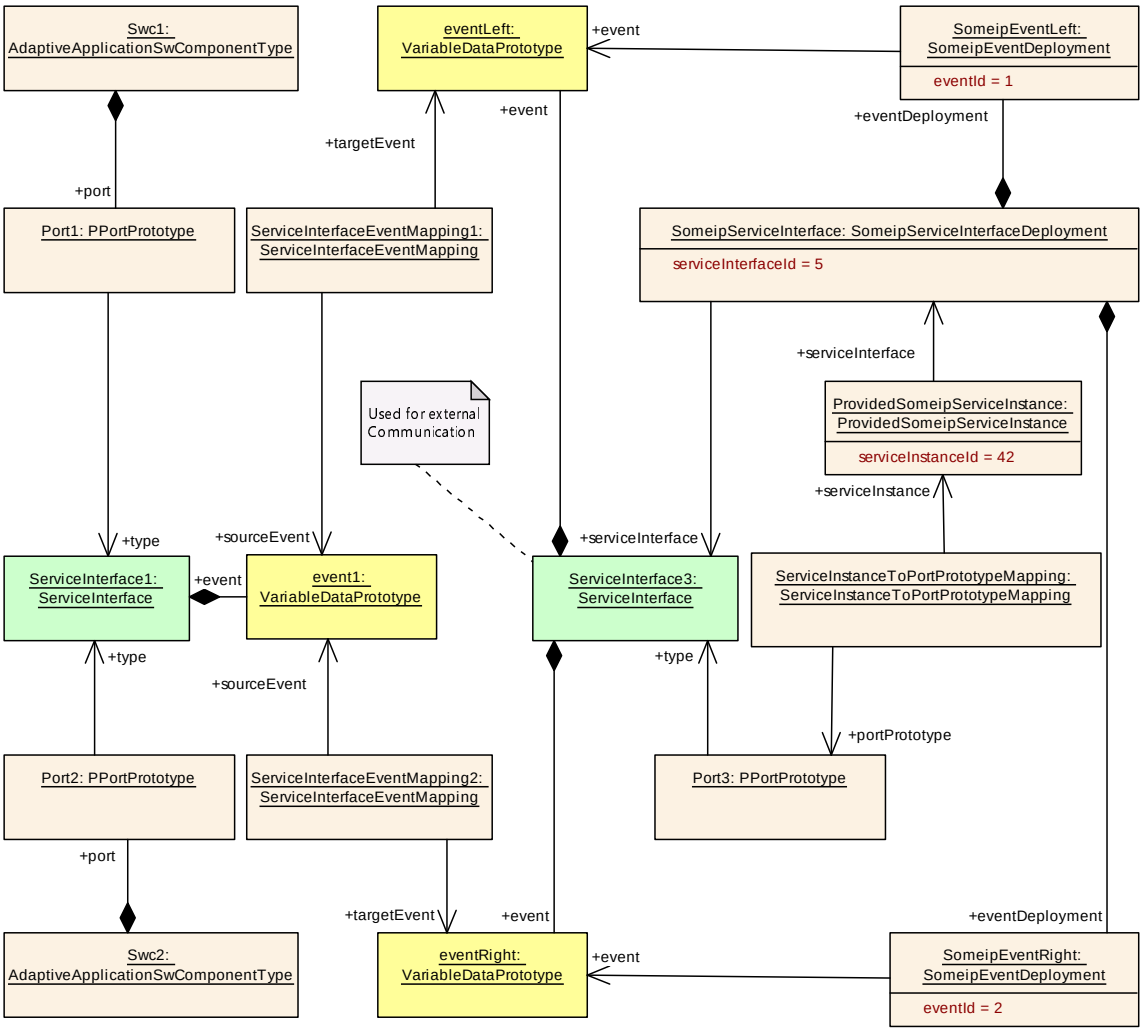


Figure A.5: Example for the deployment of a service in the presence of a `ServiceInterfaceEventMapping` to the same source `ServiceInterface`

Again, this scenario **requires** the existence of appropriately configured `ServiceInterfaceElementMapping`s.

A.3 Definition of Startup Configuration

As already mentioned, the mode-dependent startup configuration is directly aggregated by the definition of a `Process`:

```
<PROCESS>
  <SHORT-NAME>AA1</SHORT-NAME>
  <STATE-DEPENDENT-STARTUP-CONFIGS>
    <STATE-DEPENDENT-STARTUP-CONFIG>
      <EXECUTION-DEPENDENCYS>
        <EXECUTION-DEPENDENCY>
          <PROCESS-STATE-IREF>
```

```
          <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
    DECLARATION-GROUP-PROTOTYPE">/Processes/MWC/ProcessStateMachine</CONTEXT
    -MODE-DECLARATION-GROUP-PROTOTYPE-REF>
          <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
    ModeDeclarationGroups/ProcessStateMachine/Running</TARGET-MODE-
    DECLARATION-REF>
        </PROCESS-STATE-IREF>
      </EXECUTION-DEPENDENCY>
      <EXECUTION-DEPENDENCY>
        <PROCESS-STATE-IREF>
          <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
    DECLARATION-GROUP-PROTOTYPE">/Processes/MSM/ProcessStateMachine</CONTEXT
    -MODE-DECLARATION-GROUP-PROTOTYPE-REF>
          <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
    ModeDeclarationGroups/ProcessStateMachine/Running</TARGET-MODE-
    DECLARATION-REF>
        </PROCESS-STATE-IREF>
      </EXECUTION-DEPENDENCY>
    </EXECUTION-DEPENDENCYS>
    <FUNCTION-GROUP-STATE-IREFS>
      <FUNCTION-GROUP-STATE-IREF>
        <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
    DECLARATION-GROUP-PROTOTYPE">/FunctionGroupSets/ExampleFGS/ExampleFG</
    CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF>
        <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
    ModeDeclarationGroups/ExampleFG/Driving</TARGET-MODE-DECLARATION-REF>
      </FUNCTION-GROUP-STATE-IREF>
    </FUNCTION-GROUP-STATE-IREFS>
    <RESOURCE-GROUP-REF DEST="RESOURCE-GROUP">/Machines/ExampleMachine/
    Linux/resourceGroup2</RESOURCE-GROUP-REF>
    <STARTUP-CONFIG-REF DEST="STARTUP-CONFIG">/StartupConfigSets/
    StartupConfigSet_AA/AA1_Startup</STARTUP-CONFIG-REF>
   </STATE-DEPENDENT-STARTUP-CONFIG>
  </STATE-DEPENDENT-STARTUP-CONFIGS>
</PROCESS>
```

Listing A.1: Example for the definition of the `StateDependentStartupConfig` owned by a `Process`

In this example, launch dependencies exist on two other `Process`es. Both `Process`es MWC and MSM need to be in the ProcessState “Running” before AA1 is started.

The reference `StateDependentStartupConfig.functionGroupState` refers to a `ModeDeclaration` with the `shortName` `Driving` within the state machine of the underlying `Machine`. In other words the referenced `StartupConfig` that is defined in Listing A.2 is valid if the `Machine` is in the machine state `Driving`.

```
<STARTUP-CONFIG>
  <SHORT-NAME>AA1_Startup</SHORT-NAME>
  <PROCESS-ARGUMENTS>
    <PROCESS-ARGUMENT>
      <ARGUMENT>-a</ARGUMENT>
    </PROCESS-ARGUMENT>
    <PROCESS-ARGUMENT>
      <ARGUMENT>-b</ARGUMENT>
    </PROCESS-ARGUMENT>
```

```
    <PROCESS-ARGUMENT>
      <ARGUMENT>-d</ARGUMENT>
    </PROCESS-ARGUMENT>
    <PROCESS-ARGUMENT>
      <ARGUMENT>XYZ</ARGUMENT>
    </PROCESS-ARGUMENT>
  </PROCESS-ARGUMENTS>
  <SCHEDULING-POLICY>SCHEDULING-POLICY-FIFO</SCHEDULING-POLICY>
  <SCHEDULING-PRIORITY>20</SCHEDULING-PRIORITY>
</STARTUP-CONFIG>
```

Listing A.2: Example for a `StartupConfig`

The `StateDependentStartupConfig` of the `Process` is assigned to the `ResourceGroup` named `ResourceGroup2` that is defined in the Machine Manifest.

The corresponding definition of a `Machine` contains a `OsModuleInstantiation` that in turn owns the two `ResourceGroup`s named `ResourceGroup1` and `ResourceGroup2`. This aspect can be found in Listing A.3.

```
<MACHINE>
  <SHORT-NAME>ExampleMachine</SHORT-NAME>
  <MODULE-INSTANTIATIONS>
    <OS-MODULE-INSTANTIATION>
      <SHORT-NAME>Linux</SHORT-NAME>
      <RESOURCE-GROUPS>
        <RESOURCE-GROUP>
          <SHORT-NAME>resourceGroup1</SHORT-NAME>
          <CPU-USAGE>60</CPU-USAGE>
          <MEM-USAGE>1000000</MEM-USAGE>
        </RESOURCE-GROUP>
        <RESOURCE-GROUP>
          <SHORT-NAME>resourceGroup2</SHORT-NAME>
          <CPU-USAGE>70</CPU-USAGE>
          <MEM-USAGE>2000000</MEM-USAGE>
        </RESOURCE-GROUP>
      </RESOURCE-GROUPS>
    </OS-MODULE-INSTANTIATION>
  </MODULE-INSTANTIATIONS>
</MACHINE>
```

Listing A.3: Example for the definition of a `Machine`

The example definition of a `FunctionGroupSet` is sketched in Listing A.4.

```
<FUNCTION-GROUP-SET>
  <SHORT-NAME>ExampleFGS</SHORT-NAME>
  <FUNCTION-GROUPS>
    <MODE-DECLARATION-GROUP-PROTOTYPE>
      <SHORT-NAME>ExampleFG</SHORT-NAME>
      <TYPE-TREF DEST="MODE-DECLARATION-GROUP">/ModeDeclarationGroups/
   ExampleFG</TYPE-TREF>
    </MODE-DECLARATION-GROUP-PROTOTYPE>
  </FUNCTION-GROUPS>
</FUNCTION-GROUP-SET>
```

Listing A.4: Example for the definition of a `FunctionGroupSet`

The corresponding definitions of ModeDeclarationGroups are contained in Listing A.5.

```
<AR-PACKAGE>
  <SHORT-NAME>ModeDeclarationGroups</SHORT-NAME>
  <ELEMENTS>
    <MODE-DECLARATION-GROUP>
      <SHORT-NAME>ExampleFG</SHORT-NAME>
      <INITIAL-MODE-REF DEST="MODE-DECLARATION">/ModeDeclarationGroups/
   ExampleFG/Parking</INITIAL-MODE-REF>
      <MODE-DECLARATIONS>
        <MODE-DECLARATION>
          <SHORT-NAME>Driving</SHORT-NAME>
        </MODE-DECLARATION>
        <MODE-DECLARATION>
          <SHORT-NAME>Parking</SHORT-NAME>
        </MODE-DECLARATION>
      </MODE-DECLARATIONS>
    </MODE-DECLARATION-GROUP>
    <MODE-DECLARATION-GROUP>
      <SHORT-NAME>ProcessStateMachine</SHORT-NAME>
      <INITIAL-MODE-REF DEST="MODE-DECLARATION">/ModeDeclarationGroups/
   ProcessStateMachine/Idle</INITIAL-MODE-REF>
      <MODE-DECLARATIONS>
        <MODE-DECLARATION>
          <SHORT-NAME>Idle</SHORT-NAME>
        </MODE-DECLARATION>
        <MODE-DECLARATION>
          <SHORT-NAME>Starting</SHORT-NAME>
        </MODE-DECLARATION>
        <MODE-DECLARATION>
          <SHORT-NAME>Running</SHORT-NAME>
        </MODE-DECLARATION>
        <MODE-DECLARATION>
          <SHORT-NAME>Terminating</SHORT-NAME>
        </MODE-DECLARATION>
        <MODE-DECLARATION>
          <SHORT-NAME>Terminated</SHORT-NAME>
        </MODE-DECLARATION>
      </MODE-DECLARATIONS>
    </MODE-DECLARATION-GROUP>
  </ELEMENTS>
</AR-PACKAGE>
```

Listing A.5: Example for the definition of ModeDeclarationGroups

A.4 Service Instance Mapping

This section contains some examples that explain the modeling of a mapping between a service instance and the application. The examples have been created to show both the “find” and the “offer” side of the service binding.

In the first example, depicted in Figure A.6 shows the binding of PortPrototypes to a SOME/IP-based transport layer. The left part of the diagram contains the modeling of the “find” aspect and the right part contains the modeling of the “offer” aspect.

Please note that the shortNames of the two affected PortPrototypes are different. In other words, the shortNames of the PortPrototypes are not used as a way to identify the opposite end of the service binding.

Instead, the existence of a ServiceInstanceToPortPrototypeMapping that maps a PortPrototype to a ProvidedSomeipServiceInstance or RequiredSomeipServiceInstance with the **identical value** of attribute serviceInstanceId creates the actual binding between the “find” and the “offer” end.

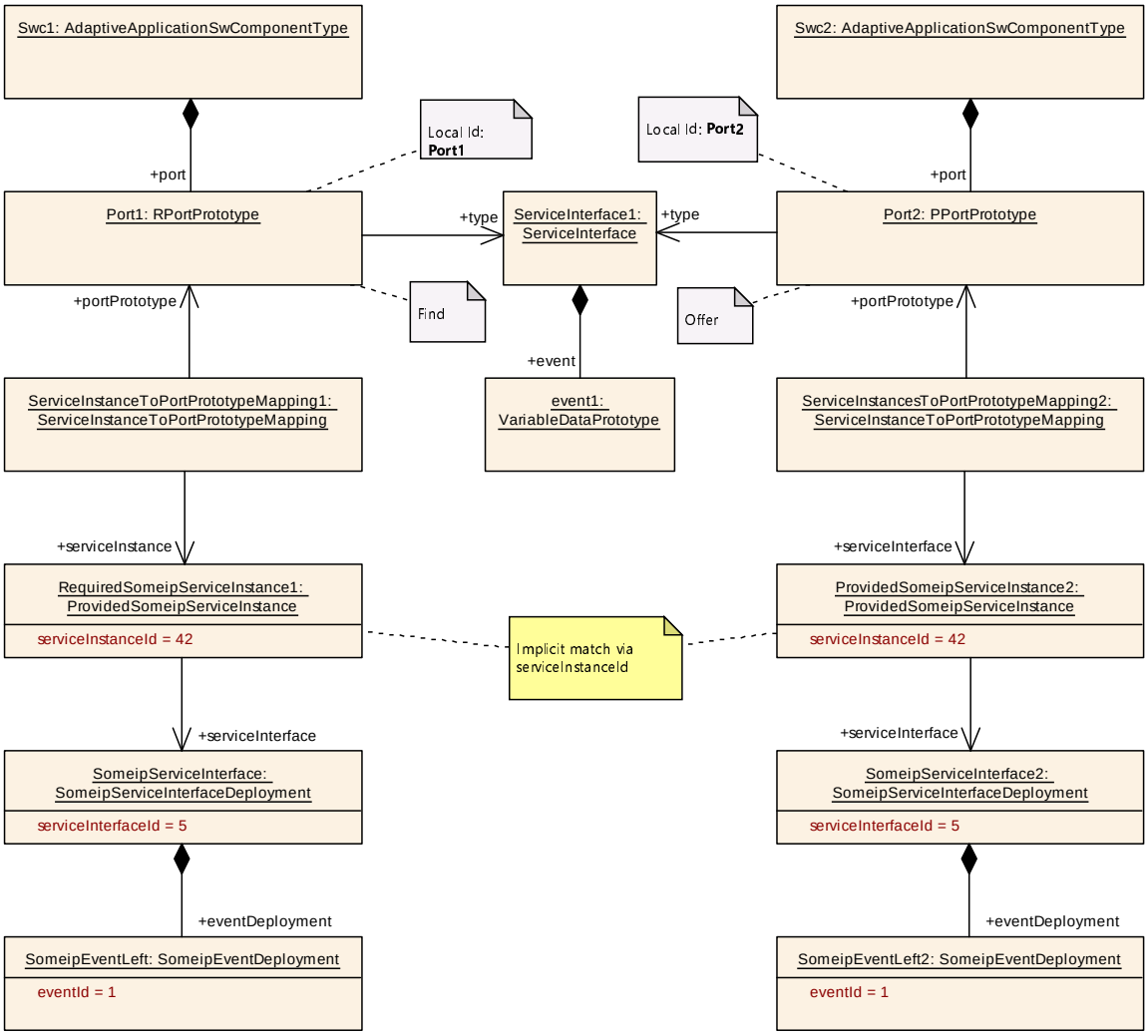


Figure A.6: Port-based binding of a service instance to the application using SOME/IP

The next example (depicted in Figure A.7) shows a binding of PortPrototypes to a user-defined transport layer. The left part of the diagram contains the modeling of the “find” aspect and the right part contains the modeling of the “offer” aspect.

Because the binding is user-defined, there are no attributes modeled on the level of the meta-model available to identify an instance according to the user-defined service

implementation. There is just no way to define attributes that are “needed anyway” for a user-defined binding.

Therefore, the only option in this case it the usage of `AdminData`, `Sdg`, and `Sd` to define an identification of the user-defined transport layer.

In order to support the comparison to the example depicted in Figure A.6, the example described in Figure A.7 uses a simple identification based on a numerical value. Again, this is an arbitrary scenario created just for the sake of explanation.

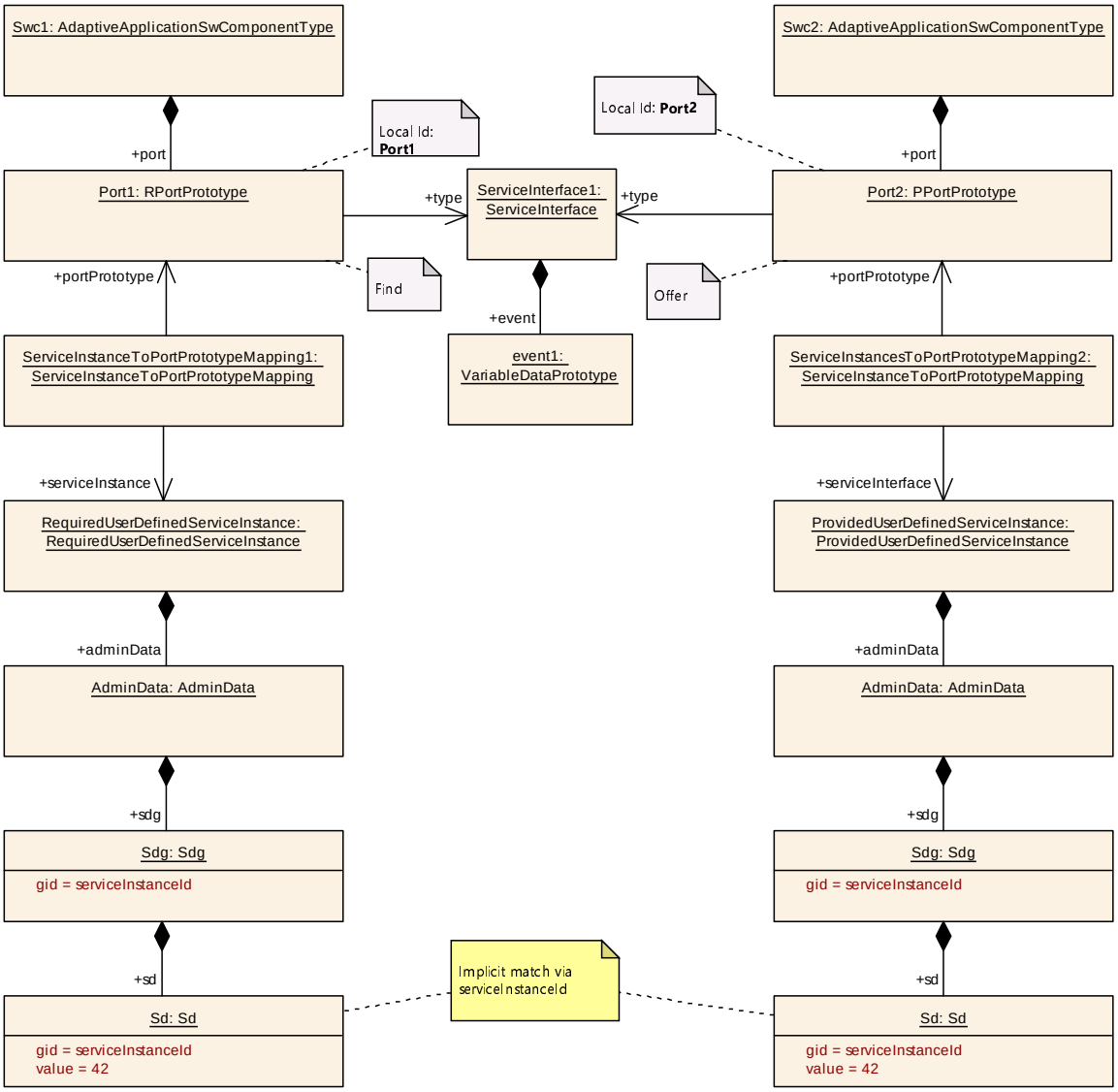


Figure A.7: Port-based binding of a service instance to the application using a user-defined binding

A.5 Radar and Camera ServiceInterface example

The example in figure A.8 shows a *Radar* `ServiceInterface` with a *BrakeEvent* and two `method`s: *Calibrate* and *Adjust*. The *Camera* `ServiceInterface` shown in figure A.9 has two events: *LaneEvent* and *SpeedLimitEvent* and one *Calibrate* `method`.

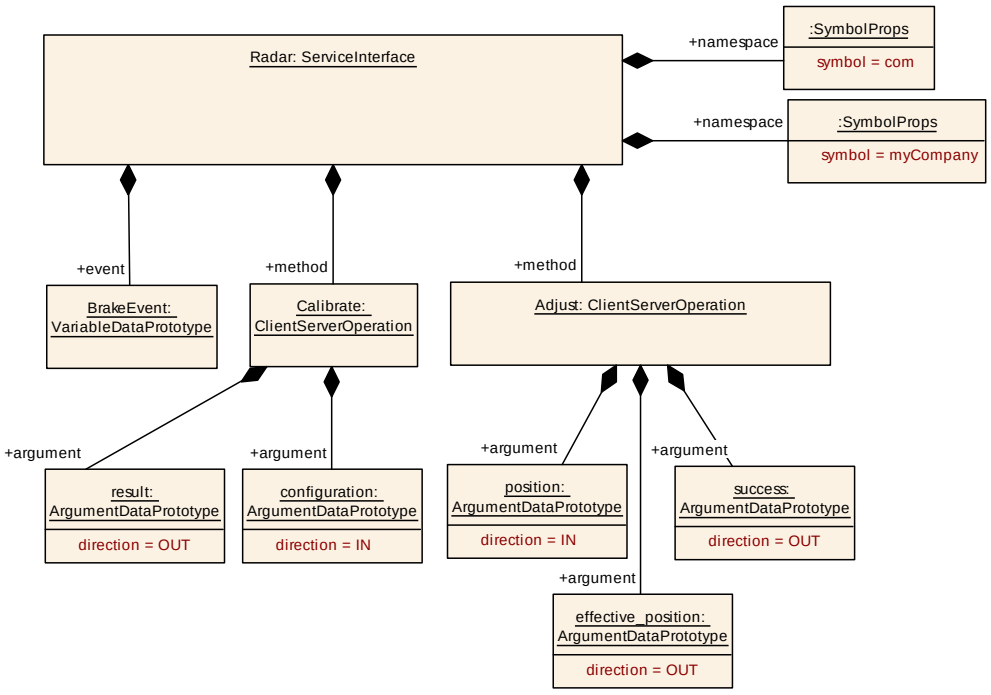


Figure A.8: Radar Service Interface

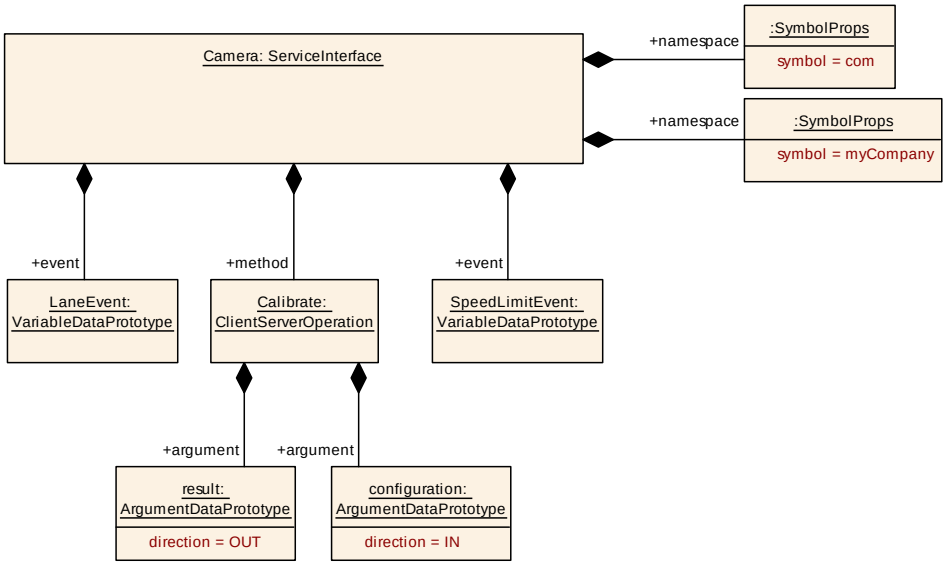


Figure A.9: Camera Service Interface

Both ServiceInterfaces *Radar* and *Camera* are mapped to a combined *RadarAndCamera* ServiceInterface with an Service Interface Element Mapping since both ServiceInterfaces have a method with the same name: *Calibrate*.

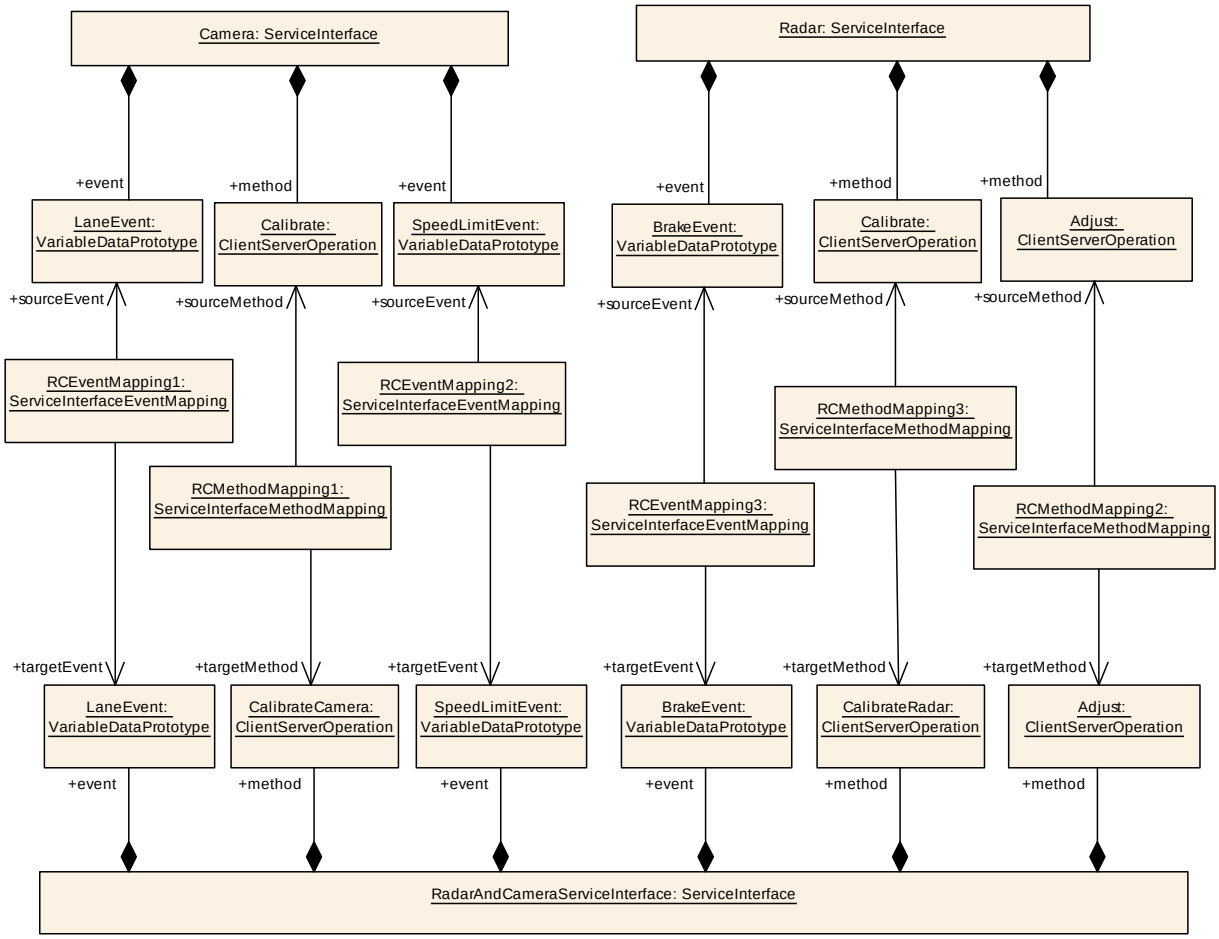


Figure A.10: Service Interface Element Mapping example

The combined ServiceInterface is offered over the network as a SOME/IP Service. Figure A.11 shows the assignment of the SOME/IP serviceInterfaceId to 31.

In addition SOME/IP eventIds are assigned to the events and methodIds are assigned to the methods. Furthermore a single SomeipEventGroup is defined to which all SomeipEventDeployments of the *RadarAndCamera* ServiceInterface are assigned.

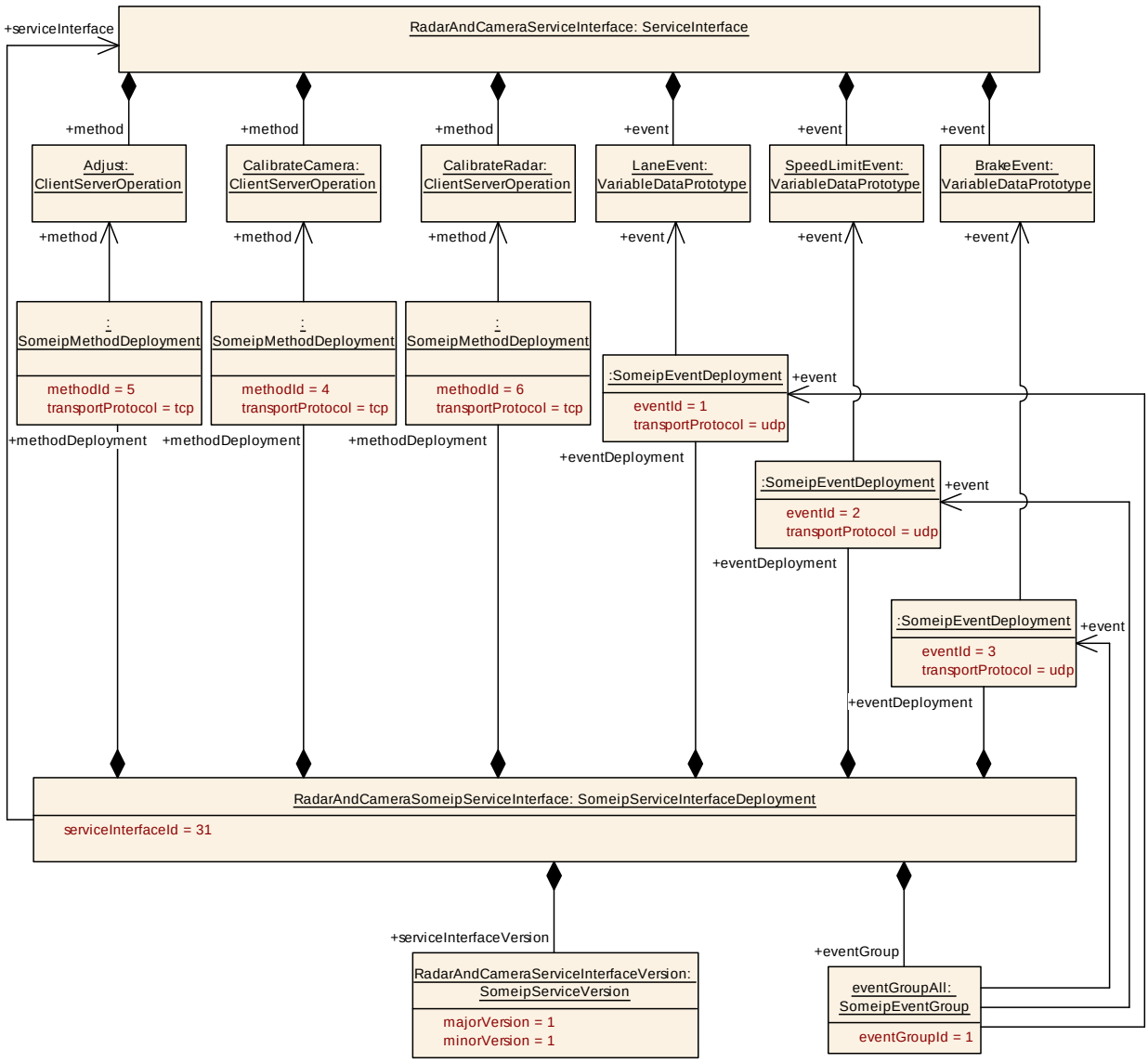


Figure A.11: SOME/IP Deployment

Figure A.12 shows a modeled `ProvidedSomeipServiceInstance` that is mapped to a `Machine`.

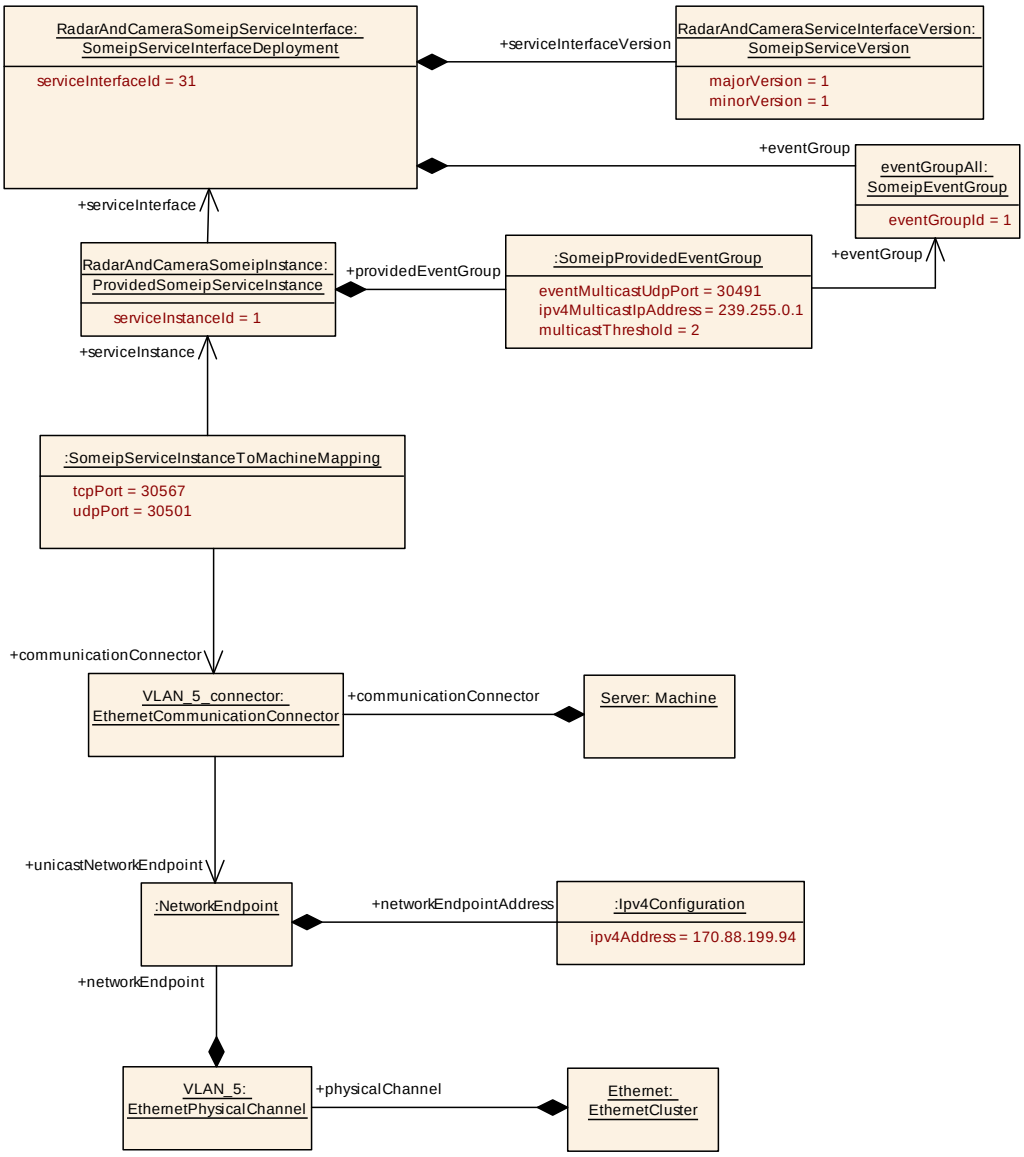


Figure A.12: SOME/IP Provided Service Instance

The displayed configuration in figure A.12 leads to a SOME/IP OfferService Message with the following content:

- ServiceId => `serviceInterfaceId` = 31
- InstanceId => `serviceInstanceId` = 1
- MajorVersion => 1
- MinorVersion => 1
- TTL => 3
- IPv4 Endpoint Option with IPv4 Address (170.88.199.94), Protocol (TCP), Port-Number (30567)

- IPv4 Endpoint Option with IPv4 Address (170.88.199.94), Protocol (UDP), Port-Number (30501)
- IP Multicast Endpoint Option with IPv4 Address (239.255.0.1), Protocol (UDP), PortNumber (30502)

An example of a `RequiredSomeipServiceInstance` is shown in Figure A.13.

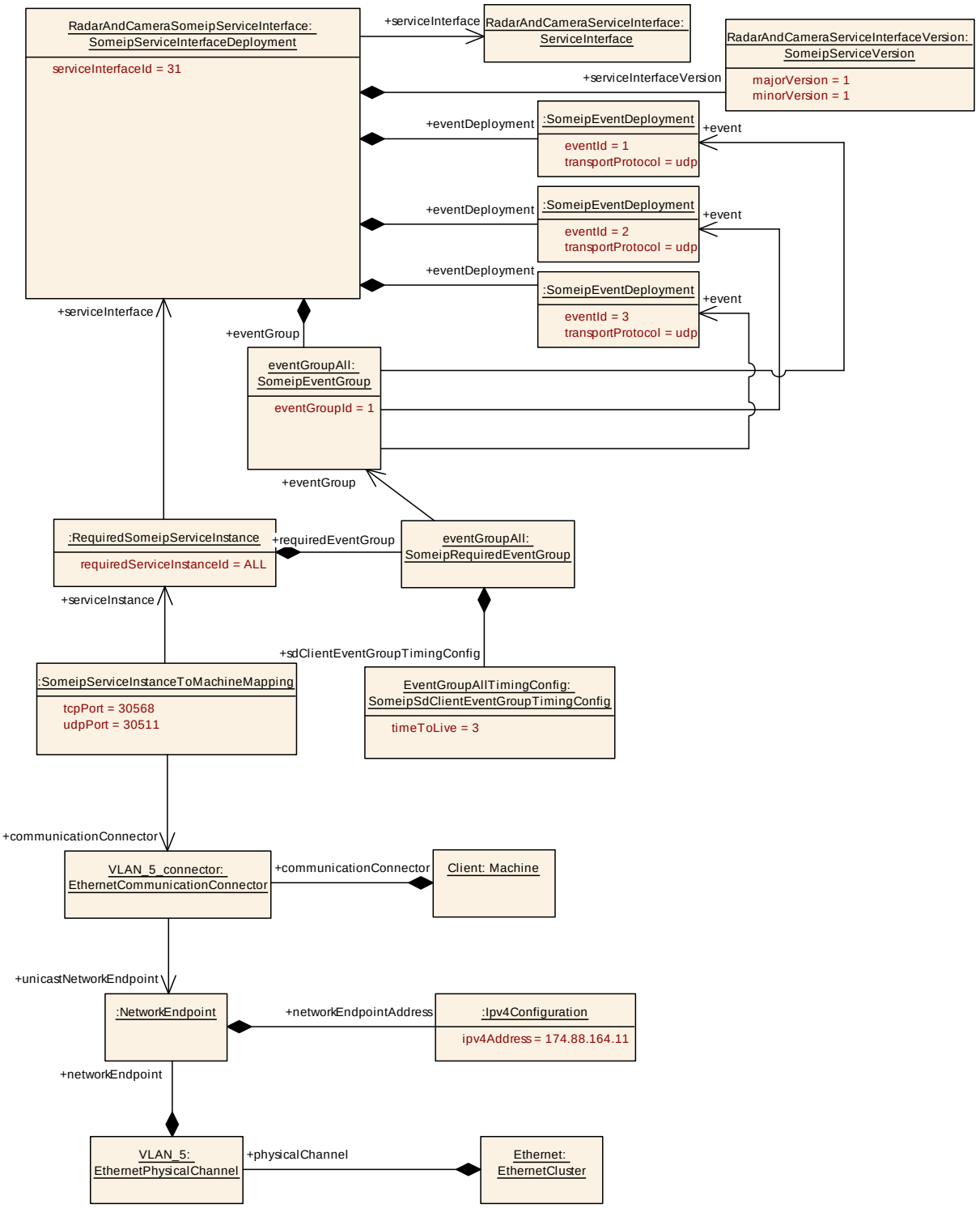


Figure A.13: SOME/IP Required Service Instance

The displayed configuration in figure A.13 leads to a SOME/IP Find Service Message with the following content:

- ServiceId => `serviceInterfaceId` = 31
- InstanceId => `RequiredSomeipServiceInstance.requiredServiceInstanceId` = ALL
- MajorVersion => `majorVersion` = 1
- MinorVersion => `minorVersion` = 1
- TTL => `RequiredSomeipServiceInstance.sdClientConfig.serviceFindTimeToLive` = 3

The displayed configuration in figure A.12 also leads to a SOME/IP SubscribeEventGroup Message content that is sent from the Service Requester to the Service Provider:

- ServiceId => taken from the OfferMessage
- InstanceId => taken from the OfferMessage
- MajorVersion => taken from the OfferMessage
- MinorVersion => taken from the OfferMessage
- Eventgroup ID => `RequiredSomeipServiceInstance.requiredEventGroup.eventGroup.eventGroupId` = 1
- TTL => `RequiredSomeipServiceInstance.requiredEventGroup.sdClientEventGroupTimingConfig.timeToLive` = 3
- IPv4 Endpoint Option with IPv4 Address (170.88.164.11), Protocol (UDP), PortNumber (30511)

A.6 Definition of Persistent Data

This chapter contains examples for the modeling of persistent data and file storage starting form the design aspect down to the definition of the persistent storage and the mapping between design and deployment.

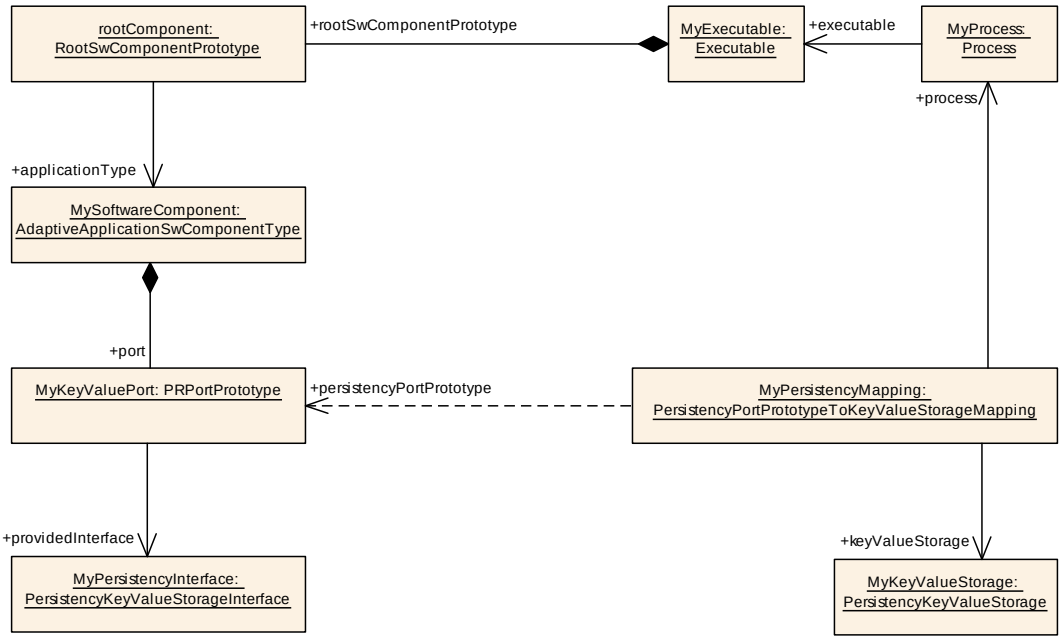


Figure A.14: Simple example modeling of persistent data (design + deployment)

The setup presented in Figure A.14 represents a case with reduced modeling of persistent data.

It is possible to extend the modeling to a deeper level of detail and also formally describe the individual data that is subject to persistency on both design and deployment level, see Figure A.15.

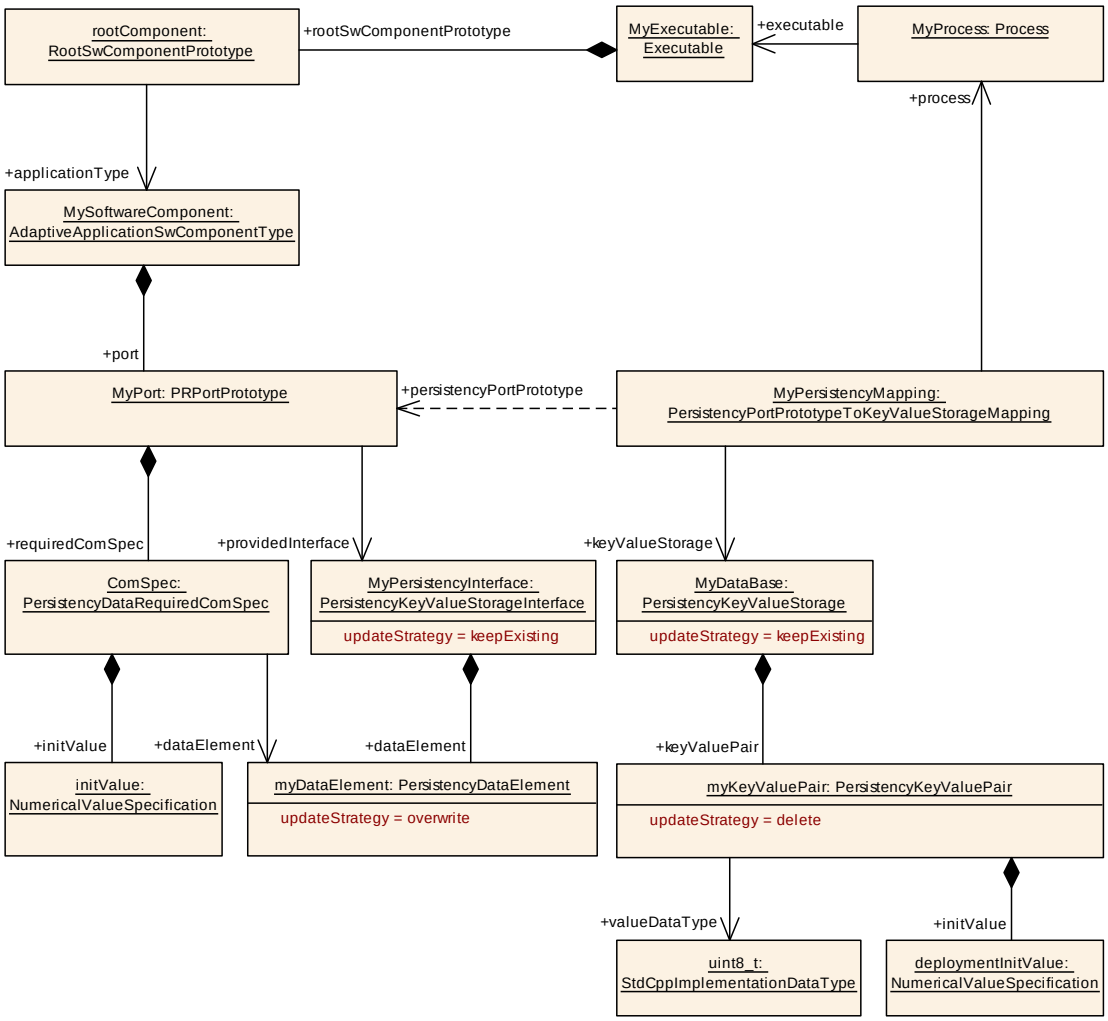


Figure A.15: Advanced example modeling of persistent data (design + deployment)

A.7 Definition of Persistent File

The setup presented in Figure A.16 represents a case with reduced modeling of persistent files.

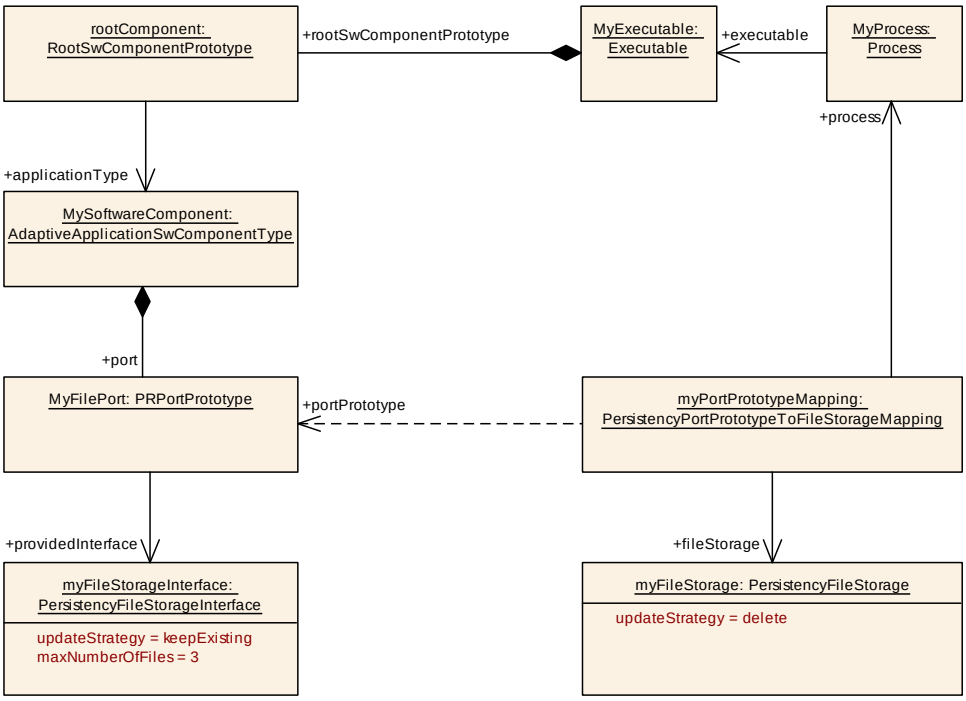


Figure A.16: Simple example modeling of persistent file (design + deployment)

It is possible to extend the modeling to a deeper level of detail and also formally describe the individual file that is subject to persistency on both design and deployment level, see Figure A.17.

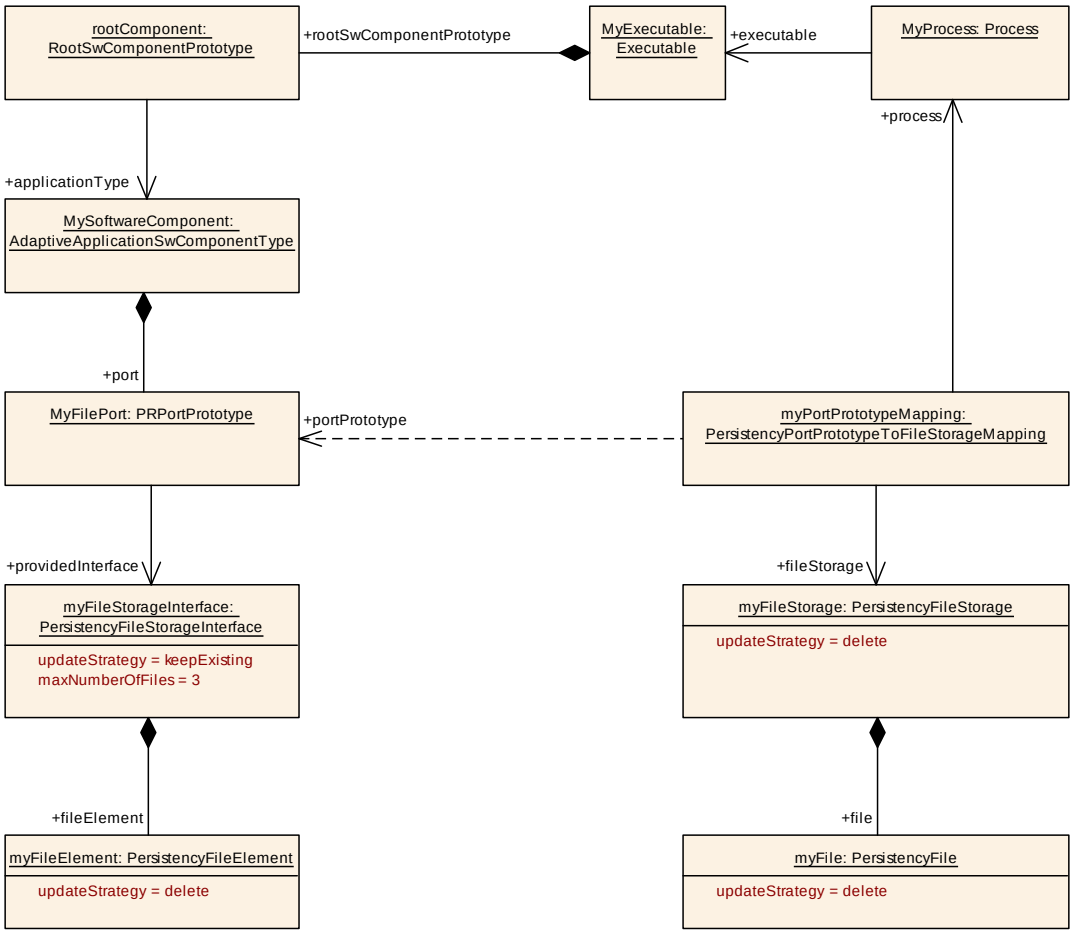


Figure A.17: Advanced example modeling of persistent file (design + deployment)

A.8 Definition of Phm interaction

This chapter contains examples for the modeling of platform health management. The example is structured into Application design and platform health management configuration.

A.8.1 Phm Application Design example

The simple example provided in figure A.18 shows the definition of a `PhmHealthChannelInterface` and a `PhmSupervisedEntityInterface`. This example will also be used in the subsequent section to define the platform health management configuration.

The `PhmHealthChannelInterface` *HealthChannel_A* defines two status attributes:

- *Good*
- *Bad*

The PhmSupervisedEntityInterface *SupervisedEntity_B* defines two checkpoints:

- *CP1*
- *CP2*

The AdaptiveApplicationSwComponentType *AdaptiveApplication* defines two RPortPrototypes

- *Hc_A* typed by *HealthChannel_A*
- *Se_B* typed by *SupervisedEntity_B*

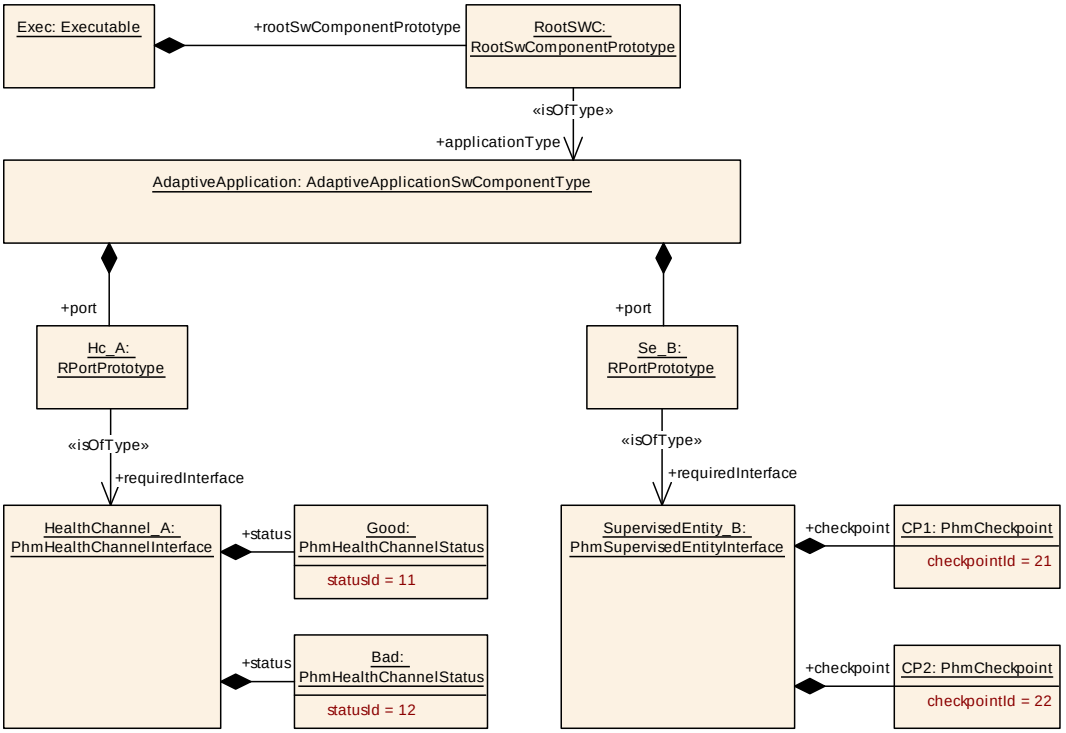


Figure A.18: Example modeling of Health Channel and Supervised Entity

A.8.2 Phm configuration example

When defining the configuration contribution for Phm it is required to first create representatives of the application design model artifacts (health channel status and supervised entity checkpoints) in the Phm configuration context. This is shown in figure A.19.

In this example the *PHM* PlatformHealthManagementContribution defines placeholder elements which refer to the respective application design model artifacts:

Example health channel:

- *Hc_Status_Good* refers to the *Good* status of *HealthChannel_A*

- *Hc_Status_Bad* refers to the *Bad* status of *HealthChannel_A*

Example supervision checkpoint:

- *Se_B_Cp1* refers to the *CP1* checkpoint of *SupervisedEntity_B*
- *Se_B_Cp2* refers to the *CP2* checkpoint of *SupervisedEntity_B*

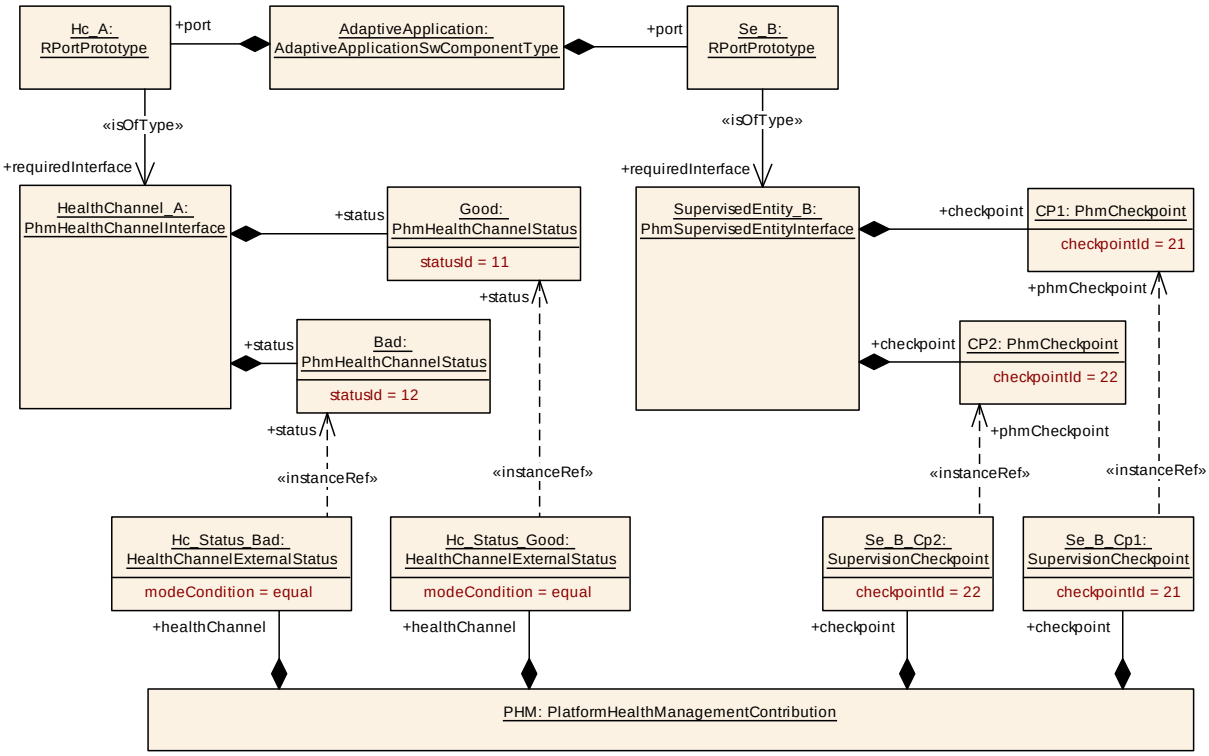


Figure A.19: Example modeling of Phm placeholder definition

Note that these instance references have a composite nature, which is shown in example figure A.20.

Here it is shown that in order to instance reference from the `HealthChannelExternalStatus` *Hc_Status_Bad* to the `PhmHealthChannelStatus` *Bad* there is the structured reference required consisting of

- `contextRootSwComponentPrototype`
- `contextRPortPrototype`
- `targetHealthChannel`

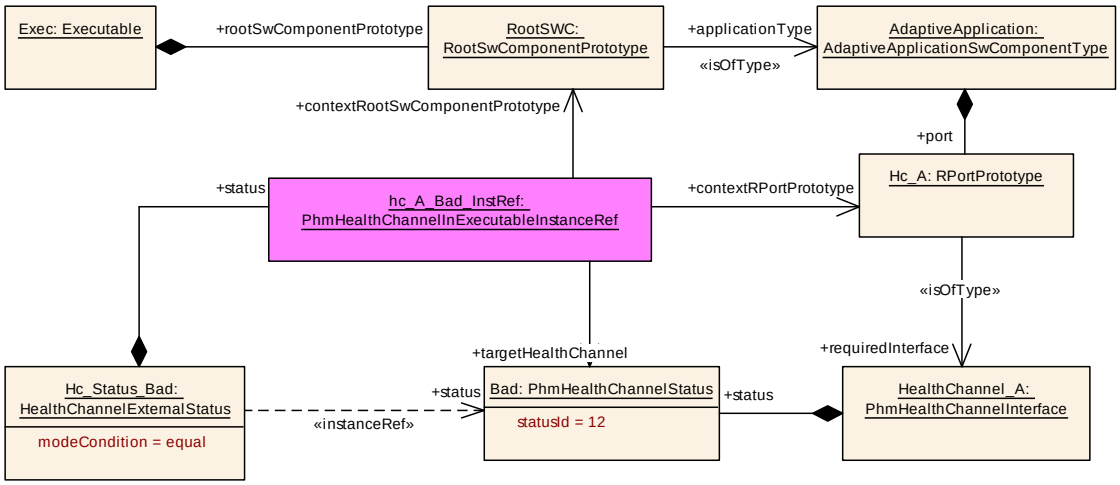


Figure A.20: Example modeling of Phm instance reference

The configuration of expressions is then based on the available placeholders of figure A.19.

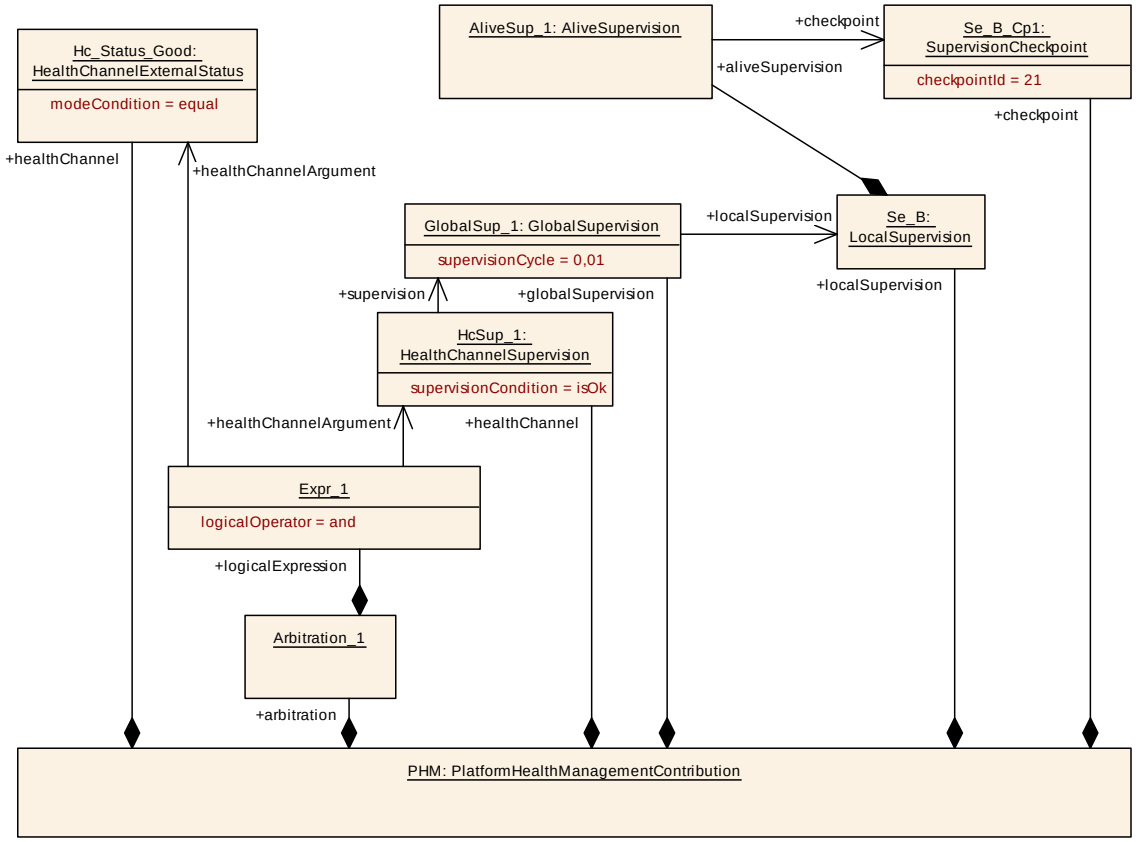


Figure A.21: Example modeling of Phm expression configuration

A.9 Scenarios to define a Vector

This section contains a non-comprehensive list of possible scenarios for the definition of a `CppImplementationDataType` of `category VECTOR`.

Please note that the general information contained in this chapter does not exclusively apply on to the vector data type. The latter has been picked as an arbitrary example for the visualization of the effect of configuration settings on the language binding.

Consequently, there is no further discussion of this topic with respect to a different kind of container data type.

Please note that for these example scenarios the namespace of a `CustomCppImplementationDataType` is assumed to be set to `x::y` and the `shortName` is assumed to be set to `CustVec`.

The `shortName` of a `StdCppImplementationDataType` is assumed to be set to `MyVec`.

If a custom `Allocator` is used in a scenario the value of `Allocator.shortName` shall be assumed to have the value `CustAlloc`.

Scenario	array size	custom allocator	custom type	Resulting C++ Code
I	No	No	No	`using MyVec = ara::core::Vector<std::uint8_t>`
II	Yes	No	No	`using MyVec = ara::core::Vector<std::uint8_t> //generator warning`
III	Yes	Yes	No	`using MyVec = ara::core::Vector<std::uint8_t, CustAlloc<std::uint8_t, MaxSize>>`
IV	No	Yes	No	`using MyVec = ara::core::Vector<std::uint8_t,CustAlloc<std::uint8_t>>`
V	Yes	Yes	Yes	`x::y::CustVec<ara::core::uint8_t, CustAlloc<std::uint8_t, MaxSize>>`
VI	Yes	No	Yes	`x::y::CustVec<std::uint8_t>`
VII	No	Yes	Yes	`x::y::CustVec<std::uint8_t, CustAlloc<std::uint8_t>>`
VIII	No	No	Yes	`x::y::CustVec<std::uint8_t>`

Table A.1: Example definitions of a `CppImplementationDataType` of `category VECTOR`

B Custom Model Extension

B.1 Overview

The AUTOSAR meta-model is obviously limited to cover the standardized part of AUTOSAR modeling. This means that the modeling stops in some cases although it is clear that the modeling capability provided by the meta-model is not sufficient to create a viable configuration.

One example for this aspect is the meta-class `ProvidedUserDefinedServiceInstance`. AUTOSAR intentionally does nothing but provide this meta-class as a basis for custom additions that configure custom transport for services.

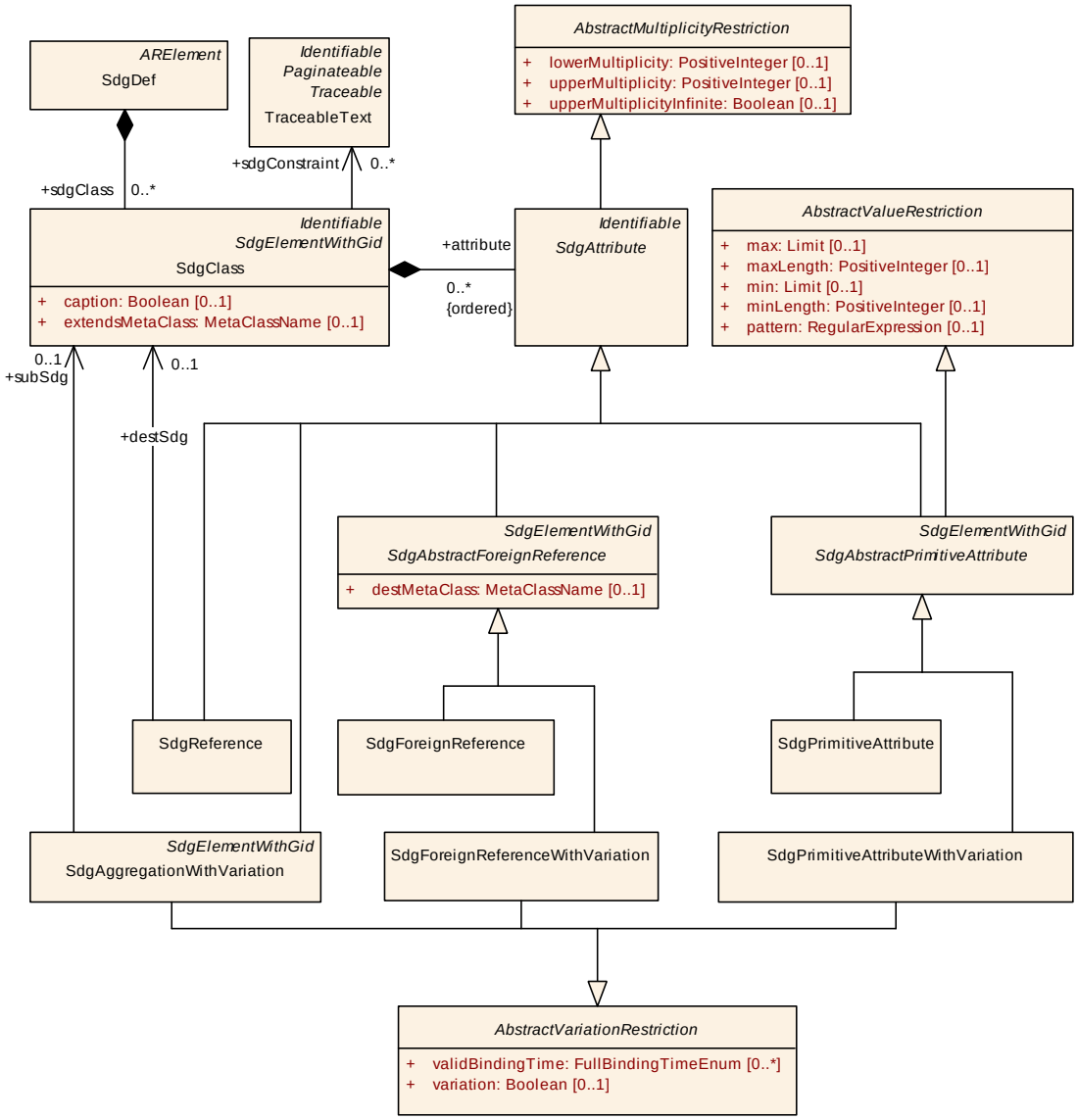


Figure B.1: Modeling of the custom model extensions

The necessary details that are required in order to actually work with a user-defined service instance are not mentioned because there are too many possibilities

of how such a ProvidedUserDefinedServiceInstance could be modeled that AUTOSAR is simply unable to cover them all.

The typical solution for such a problem is to rely on the definition of *special data groups*, formalized as Sdg. However, the direct usage of an Sdg with in a project comes with the risk that the Sdg is used slightly different in certain parts of the project.

Class	**Sdg**			
Package	M2::MSR::AsamHdo::SpecialData			
Note	Sdg (SpecialDataGroup) is a generic model which can be used to keep arbitrary information which is not explicitly modeled in the meta-model. Sdg can have various contents as defined by sdgContentsType. Special Data should only be used moderately since all elements should be defined in the meta-model. Thereby SDG should be considered as a temporary solution when no explicit model is available. If an sdg Caption is available, it is possible to establish a reference to the sdg structure.			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
gid	NameToken	1	attr	This attributes specifies an identifier. Gid comes from the SGML/XML-Term "Generic Identifier" which is the element name in XML. The role of this attribute is the same as the name of an XML - element. **Tags:**xml.attribute=true
sdgCaption	SdgCaption	0..1	aggr	This aggregation allows to assign the properties of Identifiable to the sdg. By this, a shortName etc. can be assigned to the Sdg. **Tags:**xml.sequenceOffset=20
sdgContents Type	SdgContents	0..1	aggr	This is the content of the Sdg. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

Table B.1: Sdg

It would therefore be good if there were a way to describe in terms of the AUTOSAR meta-model how a Sdg is supposed to be used.

Fortunately, this is possible by means of a corner of the meta-model that had been created for exactly this purpose: SdgClass aggregates abstract class SdgAttribute that in turn inherits to a bunch of different sub-classes.

Class	**SdgClass**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	An SdgClass specifies the name and structure of the SDG that may be used to store proprietary data in an AUTOSAR model. The SdgClass is similar to an UML stereotype.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgElementWithGid*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**SdgClass**			
attribute (ordered)	SdgAttribute	*	aggr	Defintion of the structure of the Sdg **Tags:**xml.sequenceOffset=30
caption	Boolean	0..1	attr	Specifies if a caption is required. Note: only Sdgs that have a caption can be referenced **Tags:**xml.sequenceOffset=20
extendsMeta Class	MetaClassName	0..1	attr	The AUTOSAR Meta-Class that may be extended by this SdgClass. **Tags:**xml.sequenceOffset=10
sdgConstraint	TraceableText	*	ref	Semantic constraints that restrict the structure of the special data group. **Tags:**xml.sequenceOffset=40

Table B.2: SdgClass

Class	***SdgAttribute*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	Describes the attributes of an Sdg.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*SdgAbstractForeignReference*, *SdgAbstractPrimitiveAttribute*, SdgAggregationWithVariation, Sdg Reference			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.3: SdgAttribute

B.2 Custom Attribute Definition

B.2.1 Custom Primitive Attribute Definition

In other words, `SdgClass` and `SdgAttribute` mimic the pattern found in the meta-model itself: meta-classes have attributes of different kinds.

With this mechanism it is possible to extend meta-class `ProvidedUserDefined-ServiceInstance` in order to e.g. add the ability to describe an instance Id. This chapter contains a comprehensive description of how the extension mechanism can be used to implement the instance Id.

The definition starts with an `SdgDef` that aggregates an `SdgClass` with the shortName `ProvidedUserDefinedServiceInstance`. Attribute extendsMetaClass names meta-class `ProvidedUserDefinedServiceInstance` as the subject to extension.

Class	SdgDef			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	A SdgDef groups several SdgClasses which belong to the same extension. The concept of an SdgDef is similiar to an UML Profile. **Tags:**atp.recommendedPackage=SdgDefs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
sdgClass	SdgClass	*	aggr	The owned sdgClasses which define the structure of the Sdgs **Tags:**xml.namePlural=SDG-CLASSES

Table B.4: SdgDef

The extension itself is modeled by an `SdgPrimitiveAttribute` named instanceId that is mandatory for the implementation of the user-defined service and thus has lower and upper multiplicity set to 1. The supported value interval ranges from 0..4294967295.

Class	SdgPrimitiveAttribute			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	Describes primitive special data attributes without variation. This class accepts a special data "sd" attribute.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *AbstractValueRestriction*, *Identifiable*, *Multilanguage Referrable*, *Referrable*, *SdgAbstractPrimitiveAttribute*, *SdgAttribute*, *SdgElementWithGid*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table B.5: SdgPrimitiveAttribute

```
<SDG-DEF>
  <SHORT-NAME>InstanceExtensions</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>ProvidedUserDefinedServiceInstance</SHORT-NAME>
      <GID>acme:instanceExtensions</GID>
      <EXTENDS-META-CLASS>ProvidedUserDefinedServiceInstance</EXTENDS-META-
    CLASS>
      <ATTRIBUTES>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>instanceId</SHORT-NAME>
          <CATEGORY>INTEGER</CATEGORY>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:instanceId</GID>
          <MAX>4294967295</MAX>
          <MIN>0</MIN>
        </SDG-PRIMITIVE-ATTRIBUTE>
        </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
```

```
</SDG-DEF>
```

Listing B.1: Example for the definition of a custom service instance id via SdgClass

Please note the definition of gid with value "acme:instanceExtensions" on the level of the SdgPrimitiveAttribute and the gid with value "acme:instanceId" on the level of the SdgAttribute.

The usage of the extension is summarized below. Note the usage of the gid that reflects the definition in the SdgClass and SdgPrimitiveAttribute.

Please note further that the definition of the SdgPrimitiveAttribute defines the “data type” of the

```
<PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
  <SHORT-NAME>UDSI</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="acme:instanceExtensions">
        <SD GID="acme:instanceId">42</SD>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
```

Listing B.2: Example for the specification of the value of a custom service instance id

B.2.2 Custom Complex Attribute Definition

Other extensions, e.g. using SdgAggregationWithVariation can be used to implement the aggregation of a complex attribute (that in turn may own primitive attributes or references).

In this case it makes sense to put the role of the aggregation into the value of SdgAggregationWithVariation.gid.

Note that the SdgAggregationWithVariation doesn’t aggregate further elements but refers to an SdgClass that in turn contains the attributes and references.

Class	**SdgAggregationWithVariation**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	Describes that the Sdg may contain another Sdg. The gid of the nested Sdg is defined by subSdg. Represents ’sdg’.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *AbstractVariationRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgAttribute*, *SdgElementWithGid*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
subSdg	SdgClass	0..1	ref	Supported sub Sdg Class

Table B.6: SdgAggregationWithVariation

```
<SDG-DEF>
  <SHORT-NAME>DepExt</SHORT-NAME>
```

```
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>UserDefinedServiceInterfaceDeployment</SHORT-NAME>
      <GID>acme:deploymentExtensions</GID>
      <EXTENDS-META-CLASS>UserDefinedServiceInterfaceDeployment</EXTENDS-
   META-CLASS>
      <ATTRIBUTES>
        <SDG-AGGREGATION-WITH-VARIATION>
          <SHORT-NAME>Version</SHORT-NAME>
          <GID>acme:version</GID>
          <VARIATION>false</VARIATION>
          <SUB-SDG-REF DEST="SDG-CLASS">/CustomME/DepExt/VersionInformation
   </SUB-SDG-REF>
        </SDG-AGGREGATION-WITH-VARIATION>
      </ATTRIBUTES>
    </SDG-CLASS>
    <SDG-CLASS>
      <SHORT-NAME>VersionInformation</SHORT-NAME>
      <GID>acme:deploymentExtensions</GID>
      <ATTRIBUTES>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>MajorVersion</SHORT-NAME>
          <CATEGORY>INTEGER</CATEGORY>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:majorVersion</GID>
          <MAX>4294967295</MAX>
          <MIN>0</MIN>
        </SDG-PRIMITIVE-ATTRIBUTE>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>MinorVersion</SHORT-NAME>
          <CATEGORY>INTEGER</CATEGORY>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:minorVersion</GID>
          <MAX>4294967295</MAX>
          <MIN>0</MIN>
        </SDG-PRIMITIVE-ATTRIBUTE>
      </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
```

Listing B.3: Example for the definition of a complex version information `SdgClass`

The modeling of the complex aggregation is sketched in Listing B.3. an `SdgClass` with the `gid` set to “acme:deploymentExtensions” defines an attribute that is an `SdgAggregationWithVariation` with the `shortName` “Version“ that has attribute `gid` set to “acme:version”.

This means that the `SdgAggregationWithVariation` with the `shortName` “Version“ is aggregated in the role “version” (derived from the value of the `gid`) at the `SdgClass`.

The SdgAggregationWithVariation in turn references another SdgClass with the shortName set to “VersionInformation”. This SdgClass contains two SdgPrimitiveAttributes for carrying the optional major version and the minor version. The value range of both version information is from 0 to 4294967295.

The definition side of this example is sketched in Listing B.3, the respective value side is provided in Listing B.4.

```
<USER-DEFINED-SERVICE-INTERFACE-DEPLOYMENT>
  <SHORT-NAME>UDSID</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="acme:deploymentExtensions">
        <SDG GID="acme:version">
          <SD GID="acme:majorVersion">1</SD>
          <SD GID="acme:minorVersion">0</SD>
        </SDG>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</USER-DEFINED-SERVICE-INTERFACE-DEPLOYMENT>
```

Listing B.4: Example for the specification of the value of a custom version information

B.3 Custom Foreign Reference Definition

Another aspect of custom modeling is the creation of references to meta-classes derived from Referrable. For this purpose, the meta-class SdgForeignReference resp. SdgForeignReferenceWithVariation.

Class	**SdgForeignReference**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	A reference without variation support that can point to any referrable object in an AUTOSAR Model. This class accepts the special data "Sdx" reference.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgAbstractForeignReference*, *SdgAttribute*, *SdgElementWithGid*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.7: SdgForeignReference

Class	**SdgForeignReferenceWithVariation**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef
Note	A reference with variation support that can point to any referrable object in an AUTOSAR Model. This class accepts the special data "Sdxf" reference.
Base	*ARObject*, *AbstractMultiplicityRestriction*, *AbstractVariationRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgAbstractForeignReference*, *SdgAttribute*, *SdgElementWithGid*

▽

△

Class	SdgForeignReferenceWithVariation			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table B.8: SdgForeignReferenceWithVariation

The particle “foreign” in the name of these classes represents a hint that the reference’s target is situated outside the custom modeling “bubble” depicted in Figure B.1[1].

The creation of a reference to another meta-class obviously implies the specification of the role in which the reference shall be used.

In the case of the `SdgForeignReference` the role of the reference may be defined in the attribute `gid`, contributed by the inheritance from meta-class `SdgElementWithGid`.

Class	*SdgElementWithGid* (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	A special data group element with gid is an abstract element that shall have a name (gid, "Generic Identifier").			
Base	*ARObject*			
Subclasses	*SdgAbstractForeignReference*, *SdgAbstractPrimitiveAttribute*, SdgAggregationWithVariation, SdgClass			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
gid	NameToken	0..1	attr	Specifies the name that identifies the element.

Table B.9: SdgElementWithGid

In true AUTOSAR fashion, a reference should always announce the intended meta-class to which it refers to. In the case of the `SdgForeignReference`, this information can be explicitly provided by means of the attribute `destMetaClass`, inherited from `SdgAbstractForeignReference`.

Class	*SdgAbstractForeignReference* (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	An abstract reference that can point to any referrable object in an AUTOSAR Model.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgAttribute*, *SdgElementWithGid*			
Subclasses	SdgForeignReference, SdgForeignReferenceWithVariation			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
destMetaClass	MetaClassName	0..1	attr	specifies the destination meta class of the reference.

Table B.10: SdgAbstractForeignReference

The example created for the explanation of `SdgForeignReference` assumes that a `ProvidedUserDefinedServiceInstance` wants to re-use an existing configuration for SOME/IP SD.

[1] Of course, this naming is also a hat tip to the meta-class `EcucForeignReferenceDef`

For this purpose, a custom extension of the meta-class ProvidedUserDefinedServiceInstance with the ability to refer to at most one SomeipSdServerServiceInstanceConfig is created.

In particular, the extension consists of the definition of aSdgForeignReference with the gid set to the value “acme:sdServerTimeConfig” as the representation of the role of the reference is created.

The nature of the SdgForeignReference defined in this example is determined by means of the value of attribute destMetaClass, in this case “SomeipSdServerServiceInstanceConfig”.

The custom definition of the SdgForeignReference is sketched in Listing B.5.

```
<SDG-DEF>
  <SHORT-NAME>InstanceExtensions</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>ProvidedUserDefinedServiceInstance</SHORT-NAME>
      <GID>acme:instanceExtensions</GID>
      <EXTENDS-META-CLASS>ProvidedUserDefinedServiceInstance</EXTENDS-META-
    CLASS>
      <ATTRIBUTES>
        <SDG-FOREIGN-REFERENCE>
          <SHORT-NAME>SdServerTimeConfig</SHORT-NAME>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:sdServerTimeConfig</GID>
          <DEST-META-CLASS>SomeipSdServerServiceInstanceConfig</DEST-META-
    CLASS>
        </SDG-FOREIGN-REFERENCE>
      </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
```

Listing B.5: Example for the specification of a custom foreign reference

The value side of the example in Listing B.5 can be found in Listing B.6. Note that the formalization of the reference to the respective SomeipSdServerServiceInstanceConfig is implemented by means of the reference SdgContents.sdx, wrapped into an Sdg where the attribute gid is set to the role of the reference, in this case “acme:sdServerTimeConfig”.

Class	<<atpMixed>> **SdgContents**
Package	M2::MSR::AsamHdo::SpecialData
Note	This meta-class represents the possible contents of a special data group. It can be an arbitrary mix of references, of primitive special data and nested special data groups.
Base	*ARObject*

▽

△

Class	<<atpMixed>> **SdgContents**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sd	Sd	0..1	aggr	This is one particular special data element. **Tags:**xml.sequenceOffset=40
sdf	Sdf	0..1	aggr	This is one particular special data element. **Tags:**xml.sequenceOffset=60
sdg	Sdg	0..1	aggr	This aggregation allows to express nested special data groups. By this, any structure can be represented in SpeicalData. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=50
sdx	Referrable	0..1	ref	Reference to any identifiable element. This allows to use Sdg even to establish arbitrary relationships.
sdxf	Referrable	0..1	ref	Additional reference with variant support. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild

Table B.11: SdgContents

```
<PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
  <SHORT-NAME>UDSI</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="acme:instanceExtensions">
        <SDG GID="acme:sdServerTimeConfig">
          <SDX-REF DEST="SOMEIP-SD-SERVER-SERVICE-INSTANCE-CONFIG">/SD/
   MyConf</SDX-REF>
        </SDG>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
```

Listing B.6: Example for the specification of the value of a custom foreign reference

B.4 Custom Subclass Configuration

Using the mechanism of custom model extensions it is also possible to mimic the creation of custom “subclasses”.

A possible candidate for the creation of a “subclass” outside the explicitly formalized meta-model could be the `ProvidedUserDefinedServiceInstance`. The intention could, for example, be to provide a configuration for an IPC-specific “subclass”.

A simple example of how an extension that defines a “subclass” might look like is sketched in Listing B.7.

The specific approach in this case is to define an `SdgClass` to extend `ProvidedUserDefinedServiceInstance` and this extension refers - via the aggregation of

an SdgAggregationWithVariation - to another SdgClass (in this case with the shortName “IpcProvidedServiceInstanceAttributes”) where the attributes of the custom “subclass” are defined.

On the value side, the definition of the attribute values of the “subclass” are defined in a quite straight-forward manner (see Listing B.8).

Please note that the value side in this example does not reflect the definition side in terms of the usage of values of attribute gid 100%. The existence of the SdgClass with shortName “IpcProvidedServiceInstanceAttributes” is not represented on the value side.

But, on the other hand, the intended semantics of defining an instance identifier in the context of the custom model of a “ProvidedIpcServiceInstance” can be conveyed perfectly without it.

```
<SDG-DEF>
  <SHORT-NAME>SubclassConfiguration</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>ProvidedUserDefinedServiceInstance</SHORT-NAME>
      <GID>acme:providedIpcServiceInstance</GID>
      <EXTENDS-META-CLASS>ProvidedUserDefinedServiceInstance</EXTENDS-META-
    CLASS>
      <ATTRIBUTES>
        <SDG-AGGREGATION-WITH-VARIATION>
          <SHORT-NAME>IpcSubclass</SHORT-NAME>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:attributes</GID>
          <VARIATION>false</VARIATION>
          <SUB-SDG-REF DEST="SDG-CLASS">/CustomME/SubclassConfiguration/
    IpcProvidedServiceInstanceAttributes</SUB-SDG-REF>
        </SDG-AGGREGATION-WITH-VARIATION>
      </ATTRIBUTES>
    </SDG-CLASS>
    <SDG-CLASS>
      <SHORT-NAME>IpcProvidedServiceInstanceAttributes</SHORT-NAME>
      <GID>acme:ipcProvidedServiceInstanceAttributes</GID>
      <ATTRIBUTES>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>instanceId</SHORT-NAME>
          <CATEGORY>INTEGER</CATEGORY>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:instanceId</GID>
          <MAX>65535</MAX>
          <MIN>0</MIN>
        </SDG-PRIMITIVE-ATTRIBUTE>
      </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
```

Listing B.7: Example for the specification of a custom “subclass”

And since the point of the whole approach is the creation of a custom modeling anyway, the only relevant condition for the validity of such modeling is that the affected AUTOSAR tools know how to properly parse and interpret the resulting model.

```
<PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
  <SHORT-NAME>ipcProvidedSI</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="acme:providedIpcServiceInstance">
        <SDG GID="acme:attributes">
          <SD GID="acme:instanceId">3485</SD>
        </SDG>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
```

Listing B.8: Example for the specification of a service instance in an IPC “subclass”

B.5 Custom Constraints

Another aspect that can be solved by means of model customizations is the definition of model constraints that go beyond the potentially existing constraints formulated in AUTOSAR.

In the example sketched in Listing B.9 a constraint is formulated for attribute Executable.minimumTimerGranularity.

```
<SDG-DEF>
  <SHORT-NAME>ModelConstraint</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>Executable</SHORT-NAME>
      <GID>acme:executableExtensions</GID>
      <EXTENDS-META-CLASS>Executable</EXTENDS-META-CLASS>
      <ATTRIBUTES>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>minimumTimerGranularity</SHORT-NAME>
          <CATEGORY>FLOAT</CATEGORY>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <MAX>0.5</MAX>
          <MIN>0.001</MIN>
        </SDG-PRIMITIVE-ATTRIBUTE>
      </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
```

Listing B.9: Example for the specification of a custom model constraint

On the one hand, the example expresses the expectation that the attribute Executable.minimumTimerGranularity shall exist and the second aspect of the constraint is that it shall have a value that’s between 0.001 and 0.5.

B.6 Definition of Reference from SdgClass to SdgClass

Another case that could be relevant for a custom model extension is the creation of a reference between two Sdg elements on the value side.

Semi-formal support for this scenario can be defined by means of the definition of two SdgClass elements where one defines a reference to the other.

A caveat applies. Of course, two SdgClass elements could reference each other by means of their shortName-paths (because they are derived from Identifiable).

But that's not the point, the intended reference on the value side shall exist from one Sdg (that corresponds to one of the SdgClass elements on the definition side) to another Sdg (which corresponds to the other SdgClass element on the definition side).

Sdg itself is not derived from Referrable and therefore does not have a shortName that could be used for reference building purposes.

In order to support the creation of a reference from one Sdg to another a mechanism was created using a reference from an Sdg to an SdgCaption in the role sdx.

In other words, an Sdg may aggregate an SdgCaption in the role sdgCaption. And if it does, it becomes (by extension) a valid target of a reference to this SdgCaption.

Class	**SdgCaption**			
Package	M2::MSR::AsamHdo::SpecialData			
Note	This meta-class represents the caption of a special data group. This allows to have some parts of special data as identifiable.			
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
desc	MultiLanguageOverview Paragraph	0..1	aggr	This represents a general but brief (one paragraph) description what the special data in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the special data in question.

Table B.12: SdgCaption

Therefore, the SdgClass that represents the Sdg on the target side of the reference shall define attribute sdgCaption and set it to True.

Class	**SdgReference**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	Describes an attribute of a SdgClass which is used on the definition side to model a reference from one Sdg to another Sdg on the value side.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgAttribute*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	SdgReference			
destSdg	SdgClass	0..1	ref	Refers to a SdgClass which is used on the definition side to model the destination type of the referenced Sdg. On the value side the reference is realized by means of the originating Sdg defining an sdgx attribute which refers to the sdgCaption of the referenced Sdg.

Table B.13: SdgReference

The SdgClass on the source side of the reference shall define an attribute that is actually an SdgReference to the SdgClass that represents the target Sdg on the definition side by means of the reference in the role destSdg.

As a first step, the following model fragment defines a custom model extension of an IPC channel[2] (note that attribute caption is set to True):

```
<SDG-DEF>
  <SHORT-NAME>IpcChannels</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>IpcChannel</SHORT-NAME>
      <GID>acme:ipcChannels</GID>
      <CAPTION>true</CAPTION>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
```

Listing B.10: Example for the specification of a custom model extension that represents an IPC channel

This IPC channel (named “MyIpcChannel”) shall be referenced from the definition of an SdgClass that is supposed to extend the ProvidedUserDefinedServiceInstance:

```
<SDG-DEF>
  <SHORT-NAME>InstanceExtensions</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>ProvidedUserDefinedServiceInstance</SHORT-NAME>
      <GID>acme:instanceExtensions</GID>
      <EXTENDS-META-CLASS>ProvidedUserDefinedServiceInstance</EXTENDS-META-
    CLASS>
      <CAPTION>false</CAPTION>
      <ATTRIBUTES>
        <SDG-REFERENCE>
          <SHORT-NAME>IpcChannelRef</SHORT-NAME>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <DEST-SDG-REF DEST="SDG-CLASS">/SdgClasses/IpcChannels/IpcChannel
    </DEST-SDG-REF>
        </SDG-REFERENCE>
      </ATTRIBUTES>
```

[2]Please note that this example represents an incomplete model that does not care about details of the actual configuration of the hypothetical IPC Channel and is entirely focused on the referencing topic.

```
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
```

Listing B.11: Example for the specification of a custom model extension for a `ProvidedUserDefinedServiceInstance` with a reference to an IPC channel

The model on the value side where the `ProvidedUserDefinedServiceInstance` references the IPC channel is sketched by the following model fragment:

```
<AR-PACKAGE>
  <SHORT-NAME>ServiceInstances</SHORT-NAME>
  <ELEMENTS>
    <PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
      <SHORT-NAME>MyService</SHORT-NAME>
      <ADMIN-DATA>
        <SDGS>
          <SDG GID="acme:ipcChannelRef">
            <SDX-REF DEST="SDG-CAPTION">/IpcChannels/IpcChannel1</SDX-REF>
          </SDG>
        </SDGS>
      </ADMIN-DATA>
    </PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
  </ELEMENTS>
</AR-PACKAGE>
<AR-PACKAGE>
  <SHORT-NAME>IpcChannels</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="acme:ipcChannel1">
        <SDG-CAPTION>
          <SHORT-NAME>IpcChannel1</SHORT-NAME>
        </SDG-CAPTION>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</AR-PACKAGE>
```

Listing B.12: Example for the specification of reference between `Sdg`

C General Modeling

This chapter has been created to explain model elements that are not directly related to specific design or deployment usage but have a more general scope. In other words, this chapter describes the structure and usage of some widely reusable modeling content.

C.1 Reference to a DataPrototype in a PortInterface

C.1.1 Reference to the inside of an `ApplicationDataType`

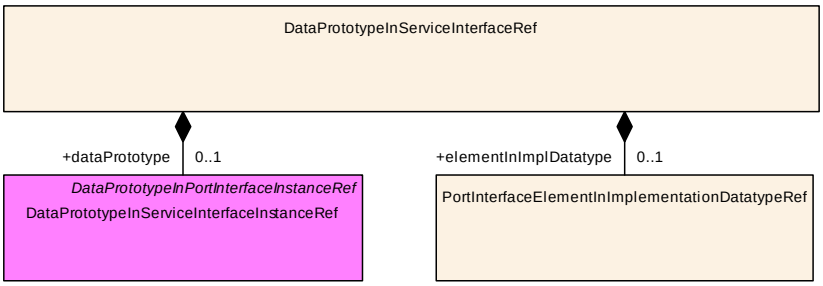


Figure C.1: Modeling of `DataPrototypeInServiceInterfaceRef`

Class	**DataPrototypeInServiceInterfaceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInPortInterfaceInstanceRef			
Note	This meta-class represents the ability to refer to an AUTOSAR DataPrototype in the context of a Service Interface. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataPrototype	DataPrototype	0..1	iref	This element represents the ability to: • refer to a DataPrototype in the context of a ServiceInterface. • refer to the internal structure of a DataPrototype in which is typed by an ApplicationDatatype the context of a ServiceInterface. **Tags:**atp.Status=draft **InstanceRef implemented by:**DataPrototypeInService InterfaceInstanceRef
elementInImpl Datatype	PortInterfaceElementIn Implementation DatatypeRef	0..1	aggr	This element represents the ability to refer to the internal structure of an AutosarDataPrototype which is typed by an ImplementationDatatype in the context of a Service Interface. **Tags:**atp.Status=draft

Table C.1: DataPrototypeInServiceInterfaceRef

Please note that the modeling of the reference to a DataPrototype in the context of a PortInterface can only be executed as the abstract template for concrete specializations because the abstract meta-class PortInterface does not aggregate a DataPrototype directly.

The abstract modeling of meta-class DataPrototypeInPortInterfaceInstanceRef is depicted in Figure C.2.

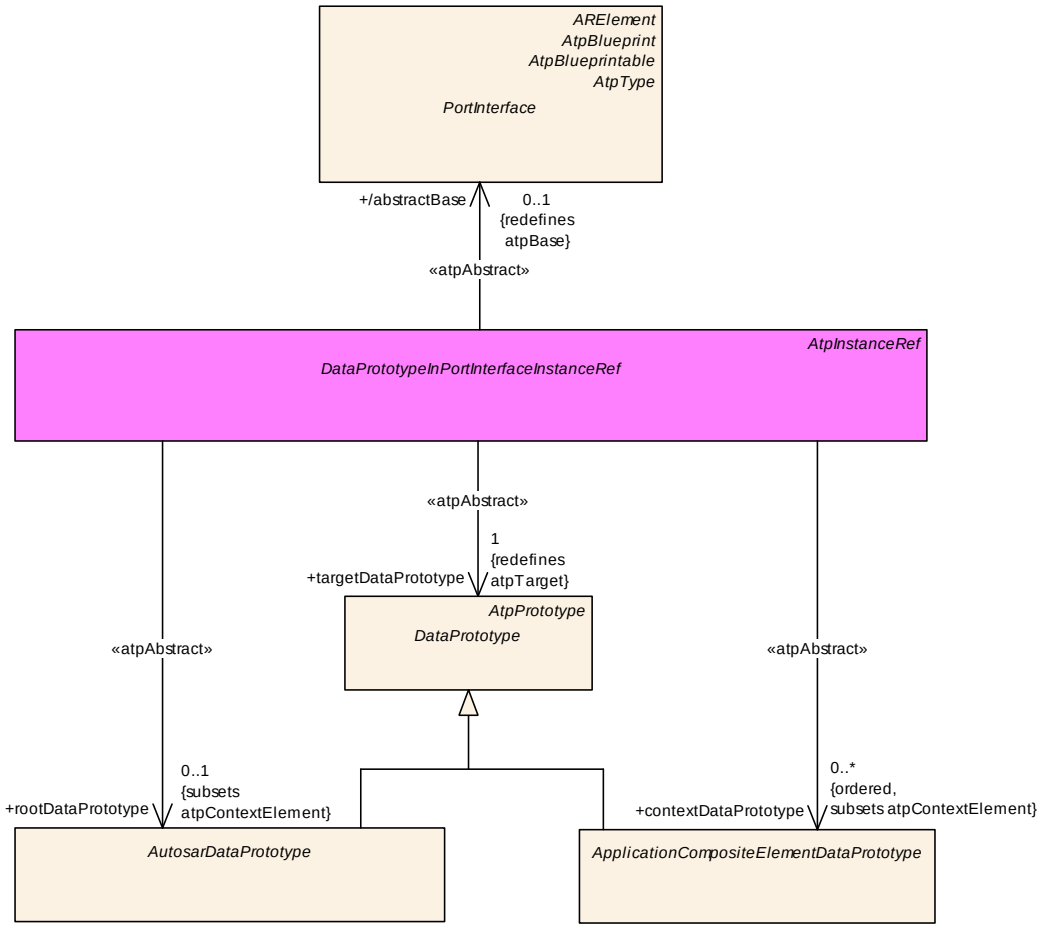


Figure C.2: Modeling of DataPrototypeInPortInterfaceInstanceRef

Class	***DataPrototypeInPortInterfaceInstanceRef*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer::InstanceRef			
Note	This meta-class represents the ability to: • refer to a DataPrototype in the context of a PortInterface. • refer to the internal structure of a DataPrototype which is typed by an ApplicationDatatype in the context of a PortInterface.			
Base	*ARObject*, *AtpInstanceRef*			
Subclasses	DataPrototypeInServiceInterfaceInstanceRef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	*DataPrototypeInPortInterfaceInstanceRef* (abstract)			
abstractBase	PortInterface	0..1	ref	**Stereotypes:** atpAbstract
contextData Prototype (ordered)	ApplicationComposite ElementDataPrototype	*	ref	**Stereotypes:** atpAbstract **Tags:**xml.sequenceOffset=20
rootData Prototype	AutosarDataPrototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:**xml.sequenceOffset=10
targetData Prototype	DataPrototype	1	ref	**Stereotypes:** atpAbstract **Tags:**xml.sequenceOffset=30

Table C.2: DataPrototypeInPortInterfaceInstanceRef

The concrete specialization for the aggregation of a `DataPrototype` in the concrete `ServiceInterface` is depicted in Figure C.3.

The meta-class `DataPrototypeInServiceInterfaceInstanceRef` inherits from `DataPrototypeInPortInterfaceInstanceRef`.

The individual references modeled in the context of `DataPrototypeInServiceInterfaceInstanceRef` specialize the abstract structure defined in the context of `DataPrototypeInPortInterfaceInstanceRef`

Figure C.3: Modeling of `DataPrototypeInServiceInterfaceInstanceRef`

Class	**DataPrototypeInServiceInterfaceInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInPortInterfaceInstanceRef			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*, *DataPrototypeInPortInterfaceInstanceRef*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	ServiceInterface	0..1	ref	**Stereotypes:** atpDerived **Tags:**atp.Status=draft
contextData Prototype (ordered)	ApplicationComposite ElementDataPrototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
rootData Prototype	AutosarDataPrototype	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10
targetData Prototype	DataPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30

Table C.3: DataPrototypeInServiceInterfaceInstanceRef

C.1.2 Reference to the inside of a `CppImplementationDataType`

Please note that the modeling of instanceRef-like references into the internals of a `CppImplementationDataType` differs from the way how internals of an `ImplementationDataType` could be referenced.

In particular, references to context elements can be directed to `ImplementationDataTypeElement` because both arrays and structures are modeled by means of `ImplementationDataTypeElement`.

This approach has changed with the advent of `CppImplementationDataType` and therefore the same approach is not possible for `CppImplementationDataTypeElement`.

In the case of `CppImplementationDataType`, both `CppImplementationDataTypeElement` and `CppImplementationDataType` can become the target of a context reference. And since the context reference is supposed to be ordered it is simply not possible to straight up model two context references, one for `CppImplementationDataType` and one for `CppImplementationDataTypeElement`.

Instead, it is necessary to introduce an abstract base class named `CppImplementationDataTypeContextTarget` for both `CppImplementationDataType` and `CppImplementationDataTypeElement` and then **direct context references at the abstract base class**.

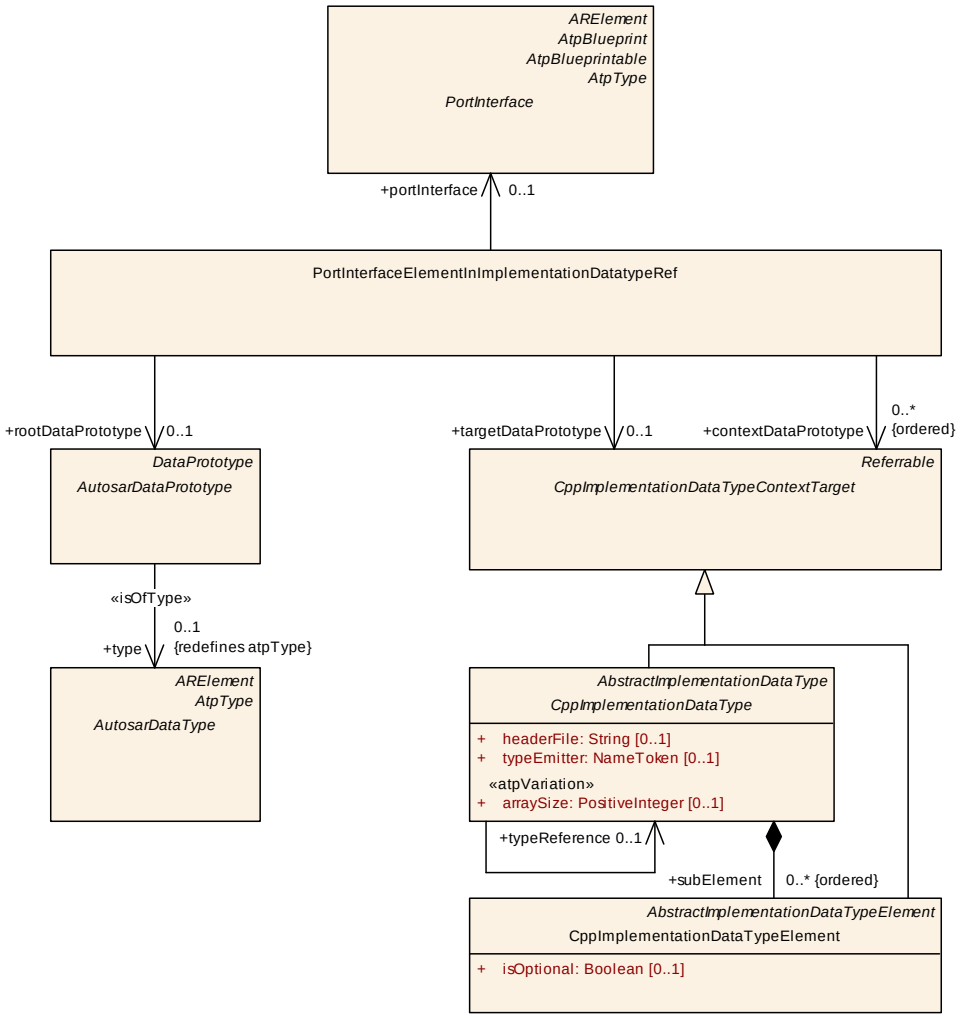


Figure C.4: Modeling of `PortInterfaceElementInImplementationDatatypeRef`

Class	**PortInterfaceElementInImplementationDatatypeRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class represents the ability to refer to the internal structure of an AutosarDataPrototype which is typed by an implementationDatatype in the context of a PortInterface. In other words, this meta-class shall not be used to model a reference to the **AutosarDataPrototype as a target itself**, **even** if the AutosarDataPrototype is typed by an ImplementationDataType **and even** if that ImplementationDataType represents a composite data type. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contextData Prototype (ordered)	CppImplementation DataTypeContextTarget	*	ref	This is a context in case there are subelements with explicit types. The reference has to be ordered to properly reflect the nested structure. **Tags:**atp.Status=draft
portInterface	PortInterface	0..1	ref	This is the PortInterface that contains the rootData Prototype. **Tags:**atp.Status=draft

△

Class	PortInterfaceElementInImplementationDatatypeRef			
rootData Prototype	AutosarDataPrototype	0..1	ref	This rootDataPrototype defines the AutosarDataPrototype in which the target can be found. **Tags:**atp.Status=draft
targetData Prototype	CppImplementation DataTypeContextTarget	0..1	ref	This is the target reference to a subElement that is defined inside of the rootDataPrototype. **Tags:**atp.Status=draft

Table C.4: PortInterfaceElementInImplementationDatatypeRef

Class	***CppImplementationDataTypeContextTarget*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class has the ability to serve as the context in instanceRef-like modeling for Cpp ImplementationDataType and CppImplementationDataTypeElement **Tags:**atp.Status=draft			
Base	*ARObject*, *Referrable*			
Subclasses	*CppImplementationDataType*, CppImplementationDataTypeElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table C.5: CppImplementationDataTypeContextTarget

C.2 Reference to a AutosarDataPrototype in an Executable

The creation of the meta-model for creating a reference to an `AutosarDataPrototype` in the context of an `Executable` is executed in a two-step approach where first an abstract structure of the reference is created.

The abstract structure is the basis for the refinement with respect to specific roles of `AutosarDataPrototype`s.

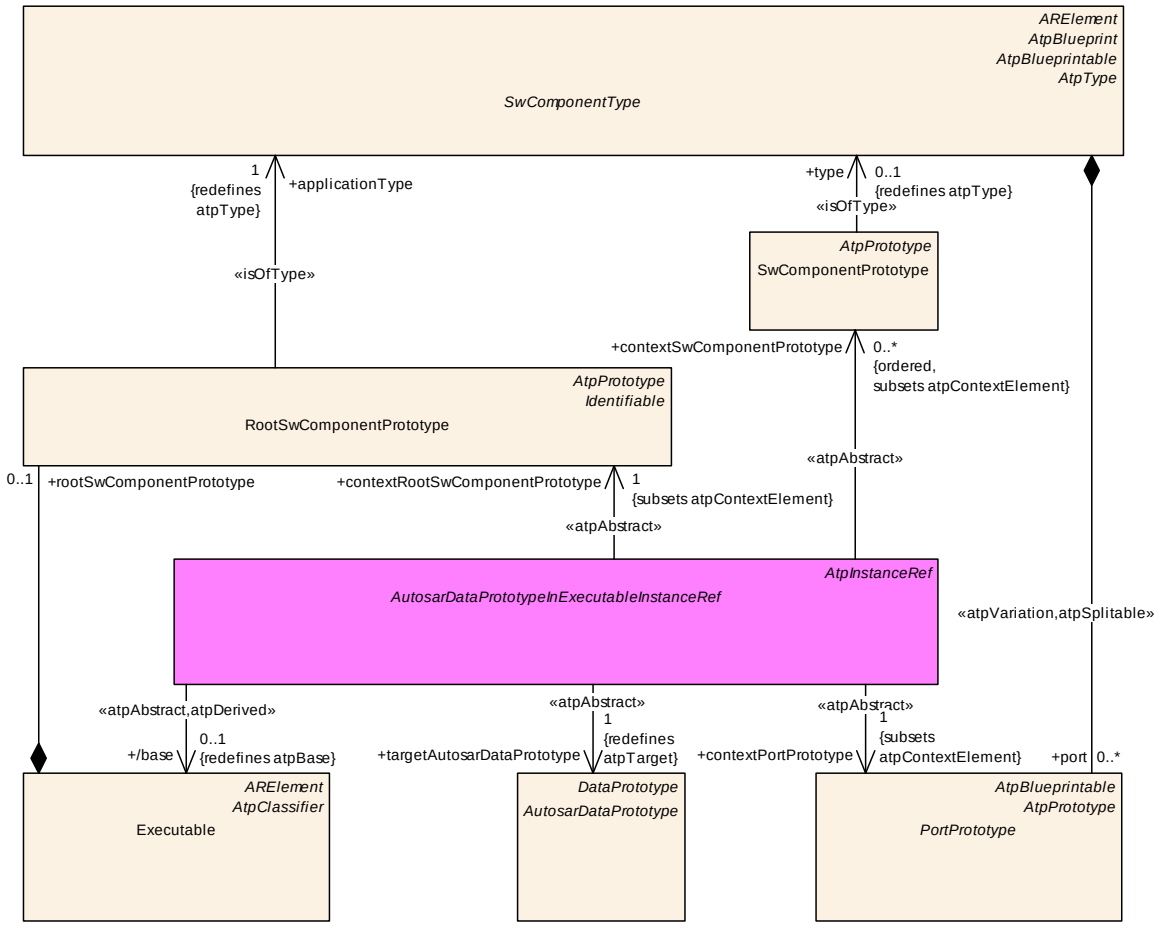


Figure C.5: Modeling of abstract `AutosarDataPrototypeInExecutableInstanceRef`

Class	***AutosarDataPrototypeInExecutableInstanceRef*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Subclasses	EventInExecutableInstanceRef, FieldInExecutableInstanceRef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpAbstract; atpDerived **Tags:**atp.Status=draft
contextPort Prototype	PortPrototype	1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=10
contextSw Component Prototype (ordered)	SwComponent Prototype	*	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=20

△

Class	***AutosarDataPrototypeInExecutableInstanceRef*** (abstract)			
targetAutosar DataPrototype	AutosarDataPrototype	1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=40

Table C.6: AutosarDataPrototypeInExecutableInstanceRef

Two specializations of `AutosarDataPrototypeInExecutableInstanceRef` exist:

- `EventInExecutableInstanceRef`
- `FieldInExecutableInstanceRef`

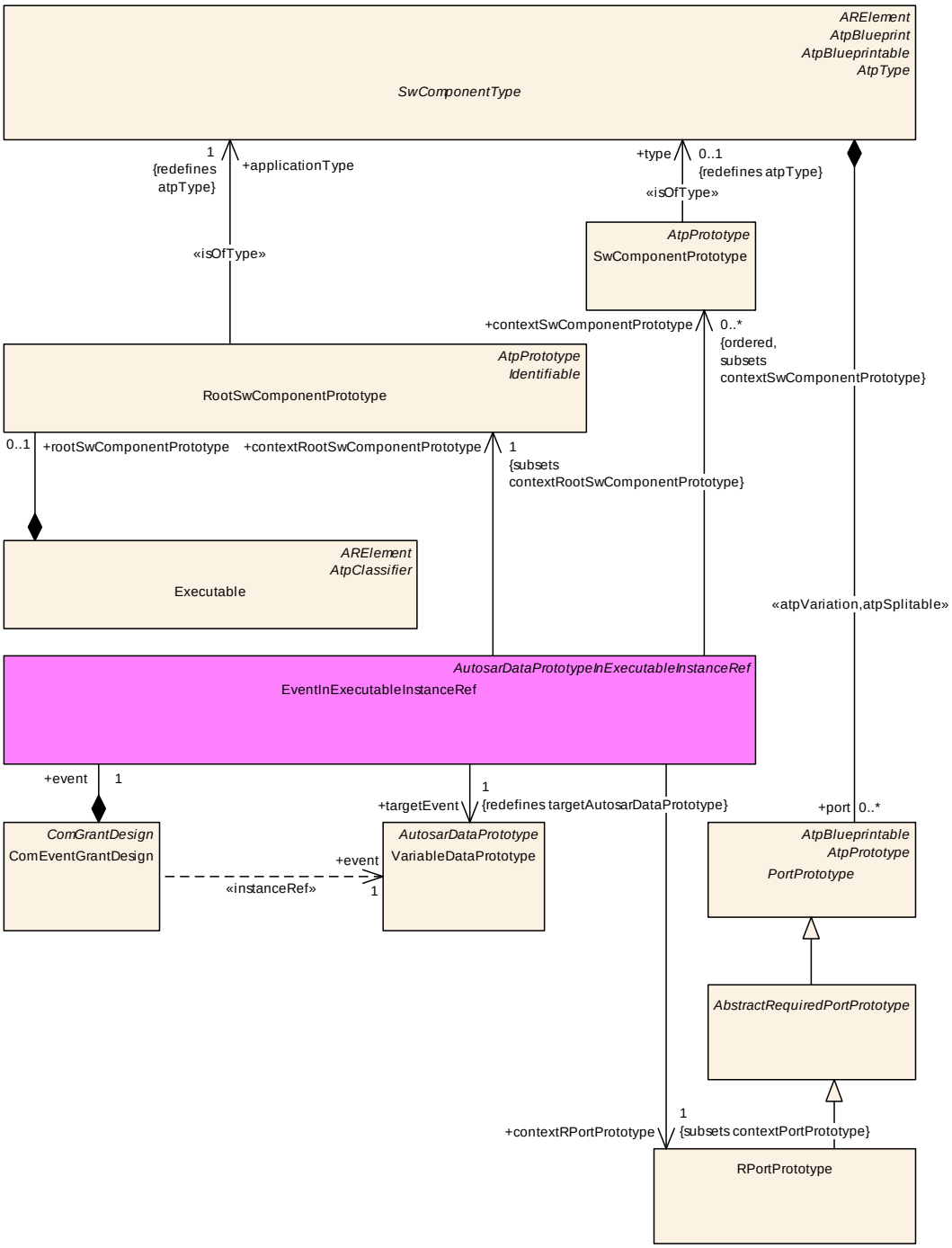


Figure C.6: Modeling of concrete `EventInExecutableInstanceRef` derived from `AutosarDataPrototypeInExecutableInstanceRef`

Class	**EventInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*, *AutosarDataPrototypeInExecutableInstanceRef*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10
contextRPort Prototype	RPortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextSw Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetEvent	VariableDataPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40

Table C.7: EventInExecutableInstanceRef

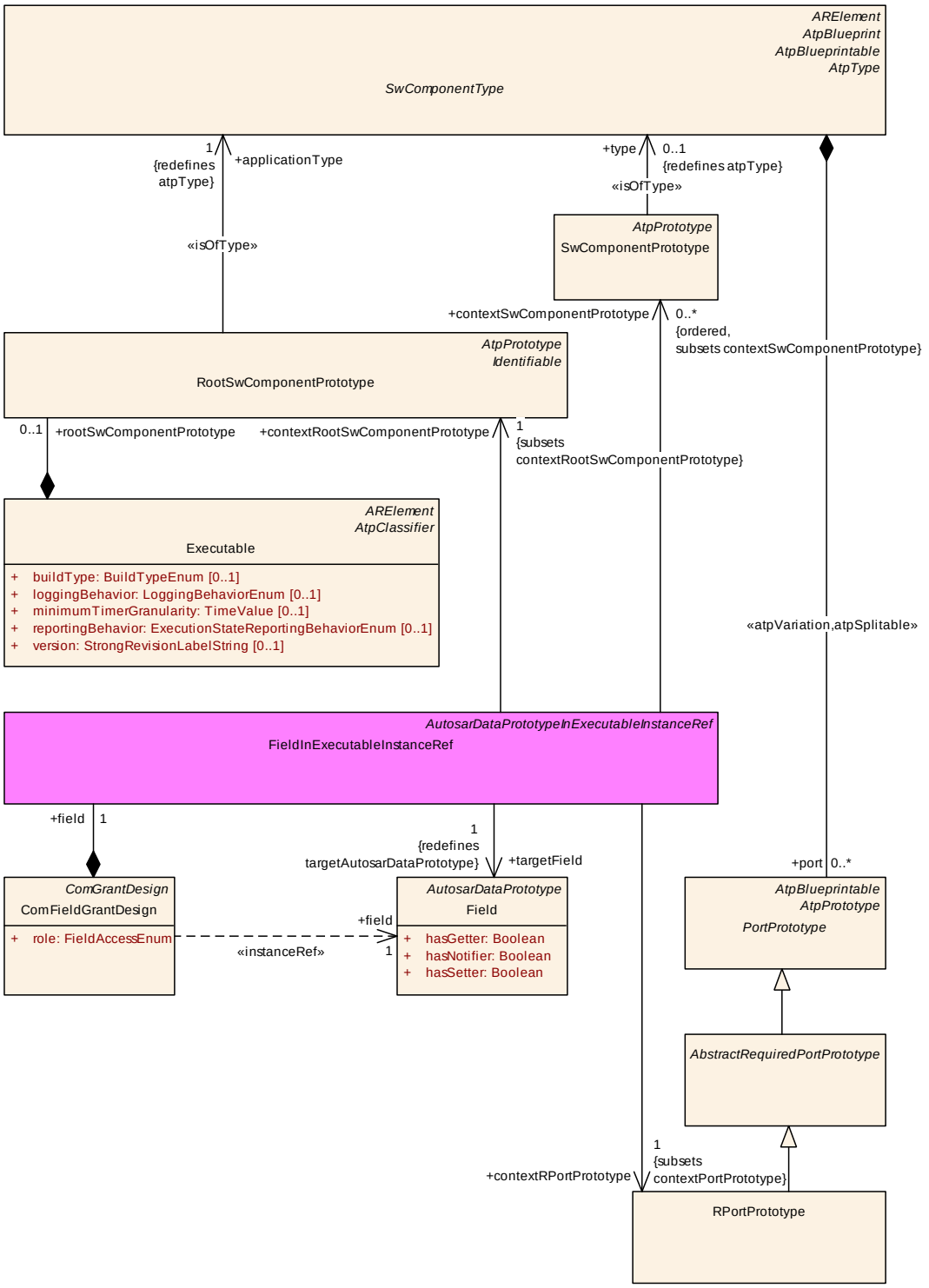


Figure C.7: Modeling of concrete `FieldInExecutableInstanceRef` derived from `AutosarDataPrototypeInExecutableInstanceRef`

Class	**FieldInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*, *AutosarDataPrototypeInExecutableInstanceRef*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10
contextRPort Prototype	RPortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextSw Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetField	Field	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40

Table C.8: FieldInExecutableInstanceRef

C.3 Reference to a PortPrototype in an Executable

The creation of the meta-model for creating a reference to a `PortPrototype` in the context of an `Executable` is executed in a two-step approach where first an abstract structure of the reference is created.

The abstract structure is the basis for the refinement with respect to specific roles of `PortPrototype`s.

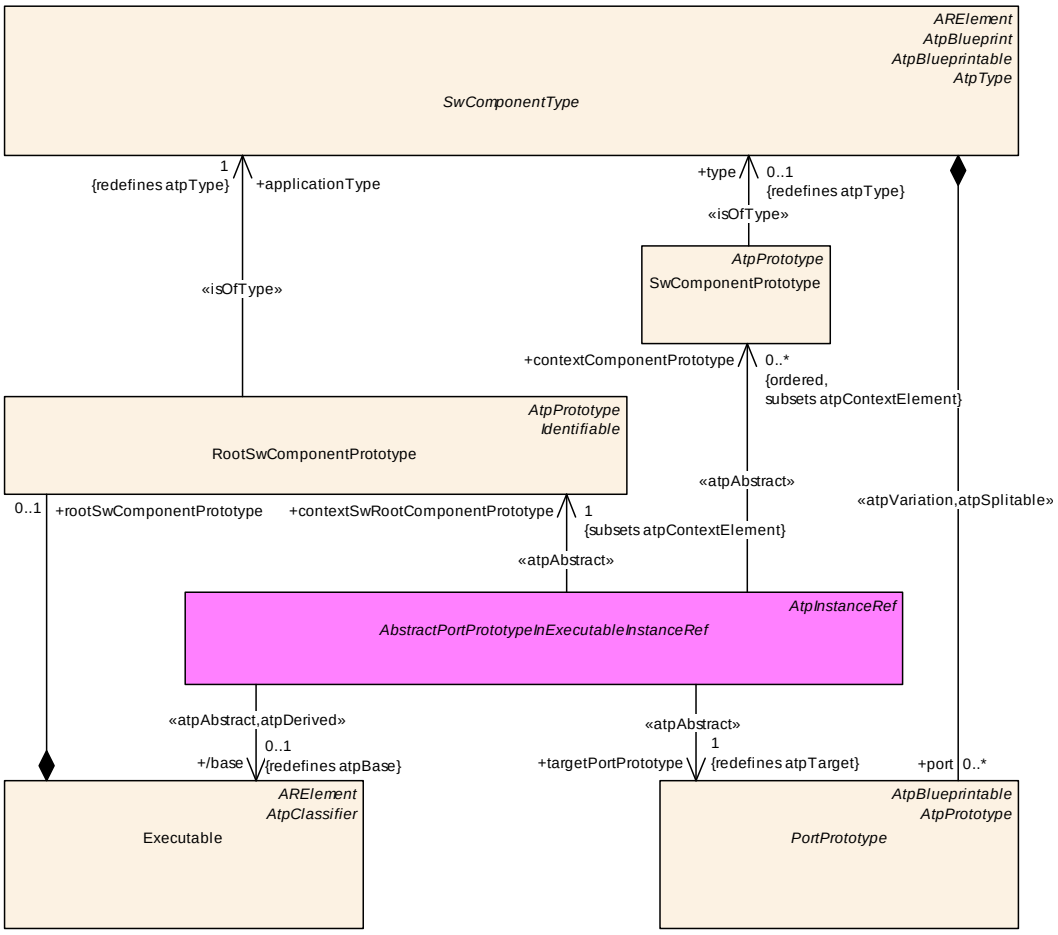


Figure C.8: Modeling of abstract `AbstractPortPrototypeInExecutableInstanceRef`

Class	***AbstractPortPrototypeInExecutableInstanceRef*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Subclasses	PPortPrototypeInExecutableInstanceRef, PortPrototypeInExecutableInstanceRef, RPortPrototypeIn ExecutableInstanceRef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpAbstract; atpDerived **Tags:**atp.Status=draft
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=20
contextSwRoot Component Prototype	RootSwComponent Prototype	1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=10

▽

△

Class	***AbstractPortPrototypeInExecutableInstanceRef*** (abstract)			
targetPort Prototype	PortPrototype	1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=30

Table C.9: AbstractPortPrototypeInExecutableInstanceRef

Three specializations of AbstractPortPrototypeInExecutableInstanceRef exist:

- PPortPrototypeInExecutableInstanceRef
- RPortPrototypeInExecutableInstanceRef
- PortPrototypeInExecutableInstanceRef

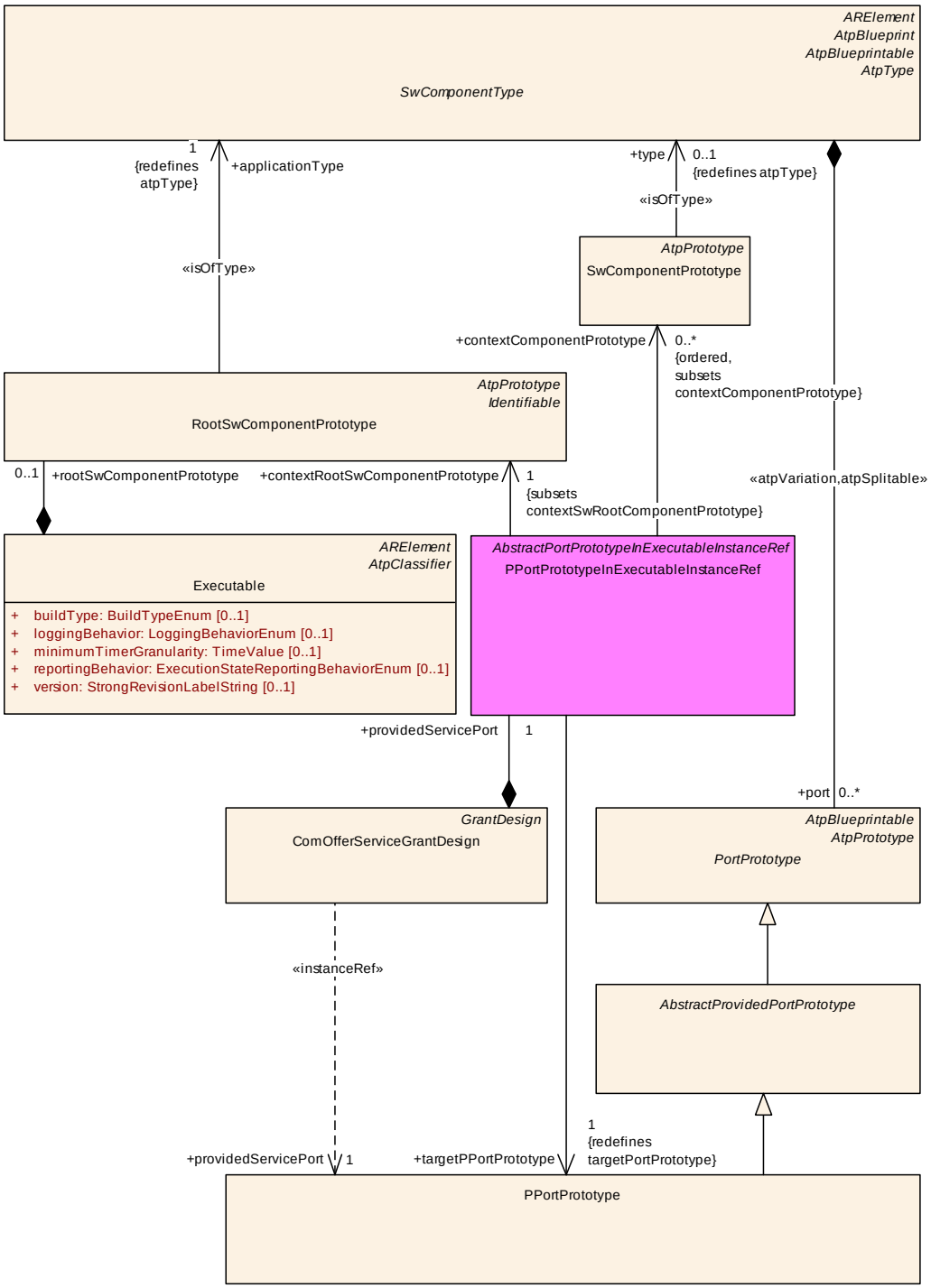


Figure C.9: Modeling of concrete `PPortPrototypeInExecutableInstanceRef` derived from `AbstractPortPrototypeInExecutableInstanceRef`

Class	**PPortPrototypeInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AbstractPortPrototypeInExecutableInstanceRef*, *AtpInstanceRef*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10
targetPPort Prototype	PPortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30

Table C.10: PPortPrototypeInExecutableInstanceRef

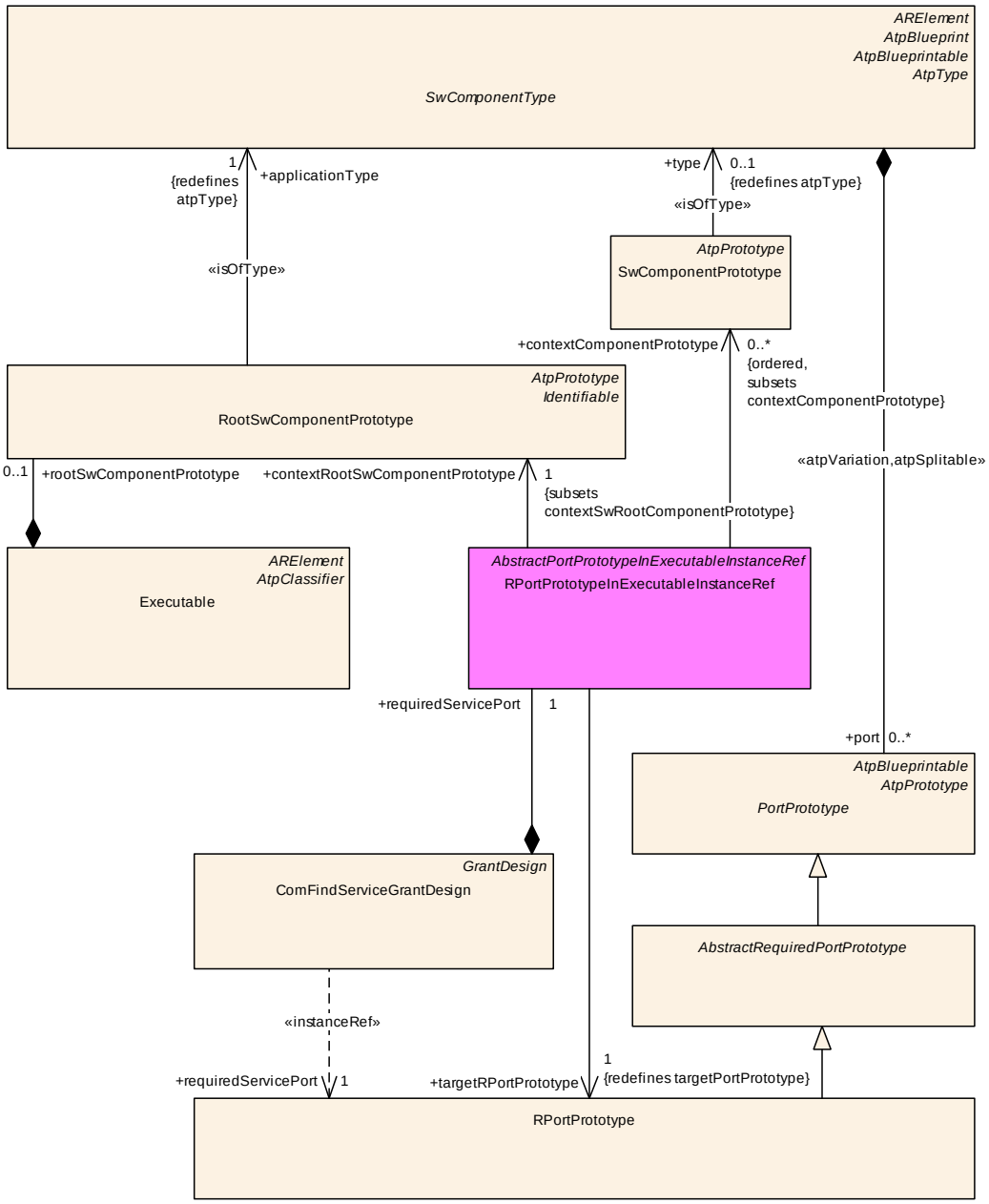


Figure C.10: Modeling of concrete `RPortPrototypeInExecutableInstanceRef` derived from `AbstractPortPrototypeInExecutableInstanceRef`

Class	**RPortPrototypeInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AbstractPortPrototypeInExecutableInstanceRef*, *AtpInstanceRef*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**RPortPrototypeInExecutableInstanceRef**			
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10
targetRPort Prototype	RPortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30

Table C.11: RPortPrototypeInExecutableInstanceRef

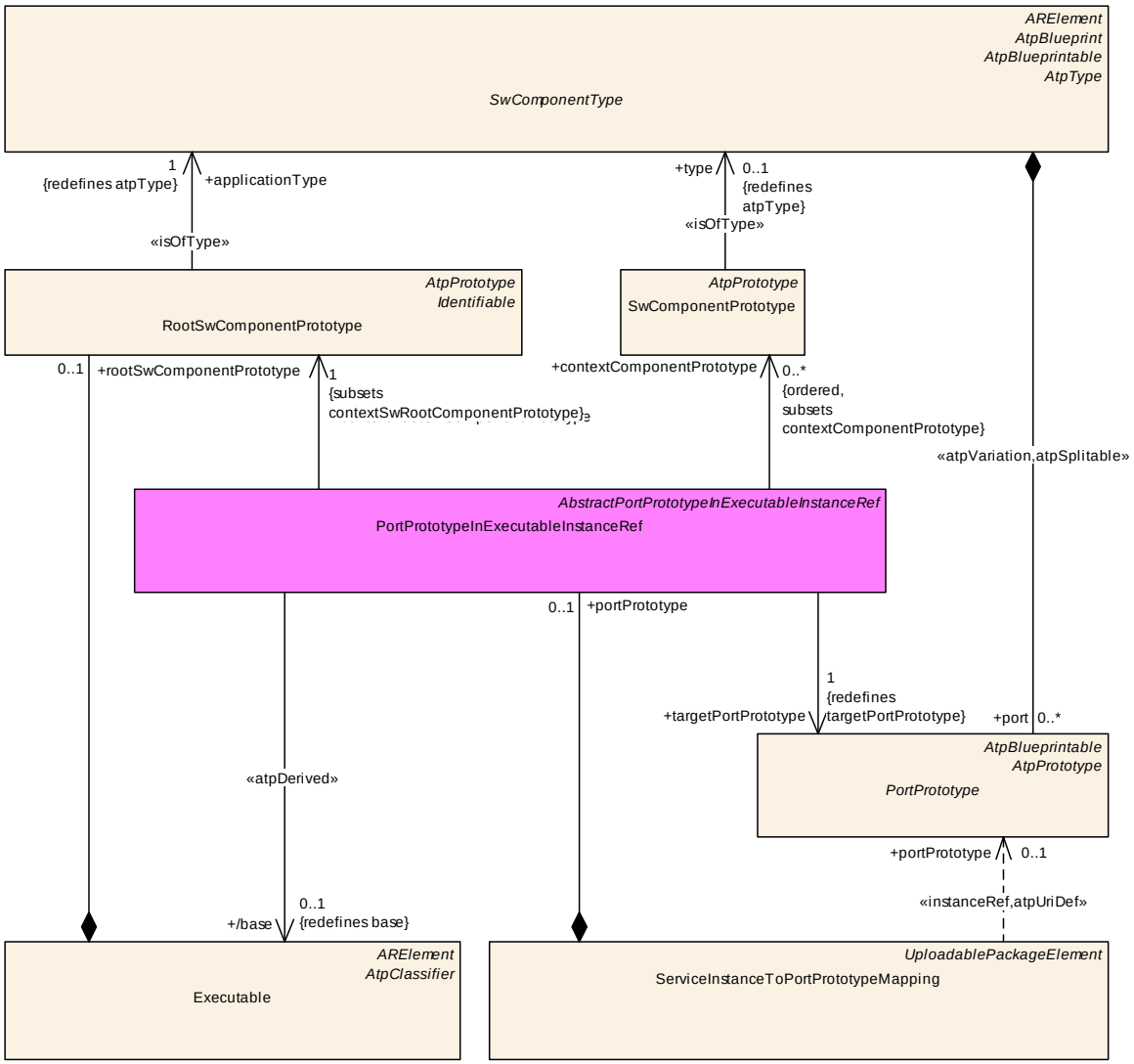


Figure C.11: Modeling of `PortPrototypeInExecutableInstanceRef`

Class	**PortPrototypeInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest::InstanceRefs			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AbstractPortPrototypeInExecutableInstanceRef*, *AtpInstanceRef*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft xml.sequenceOffset=10
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetPort Prototype	PortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40

Table C.12: PortPrototypeInExecutableInstanceRef

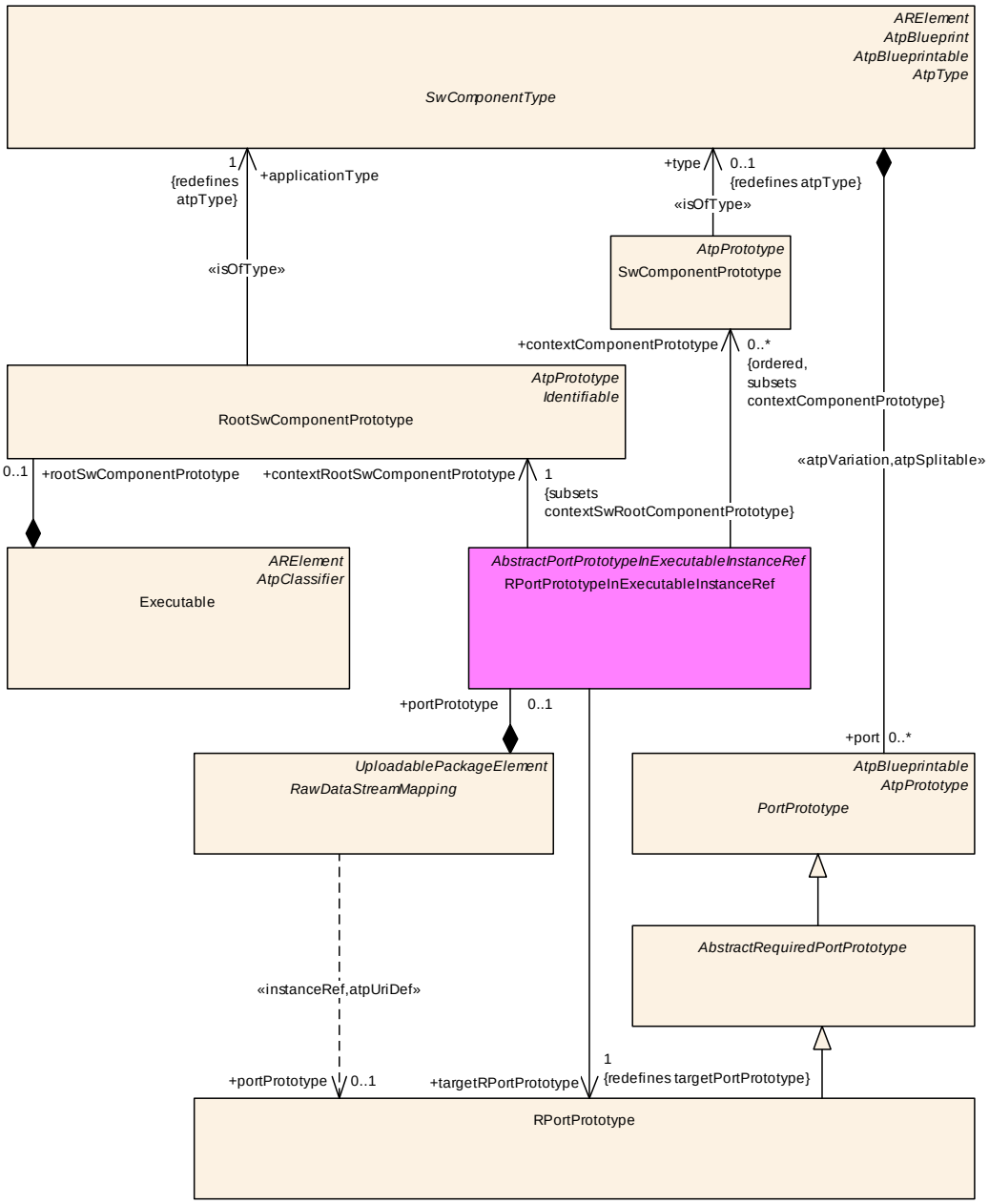


Figure C.12: Modeling of raw data streaming by means of an `RPortPrototype`

C.4 Modeling of a Method in an Executable

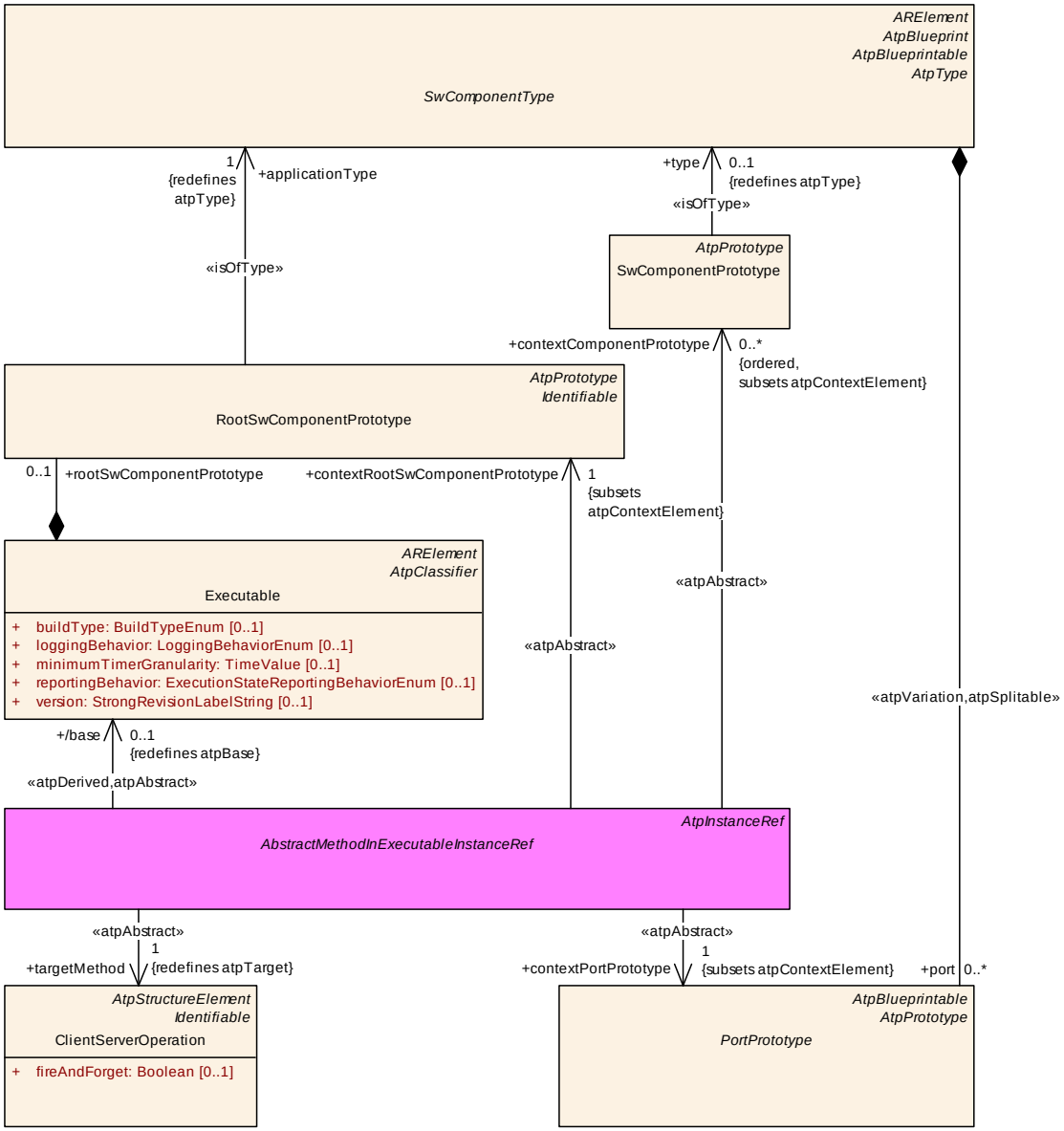


Figure C.13: Modeling of `AbstractMethodInExecutableInstanceRef`

Class	***AbstractMethodInExecutableInstanceRef*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef::MethodIn Executable			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Subclasses	ProvidedMethodInExecutableInstanceRef, RequiredMethodInExecutableInstanceRef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpAbstract; atpDerived **Tags:**atp.Status=draft

△

Class	***AbstractMethodInExecutableInstanceRef*** (abstract)			
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=20
contextPort Prototype	PortPrototype	1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=10
targetMethod	ClientServerOperation	1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=40

Table C.13: AbstractMethodInExecutableInstanceRef

C.5 Modeling of Mode-related InstanceRefs

This section illustrates the concrete modeling of the instance references used in the previous parts of this document.

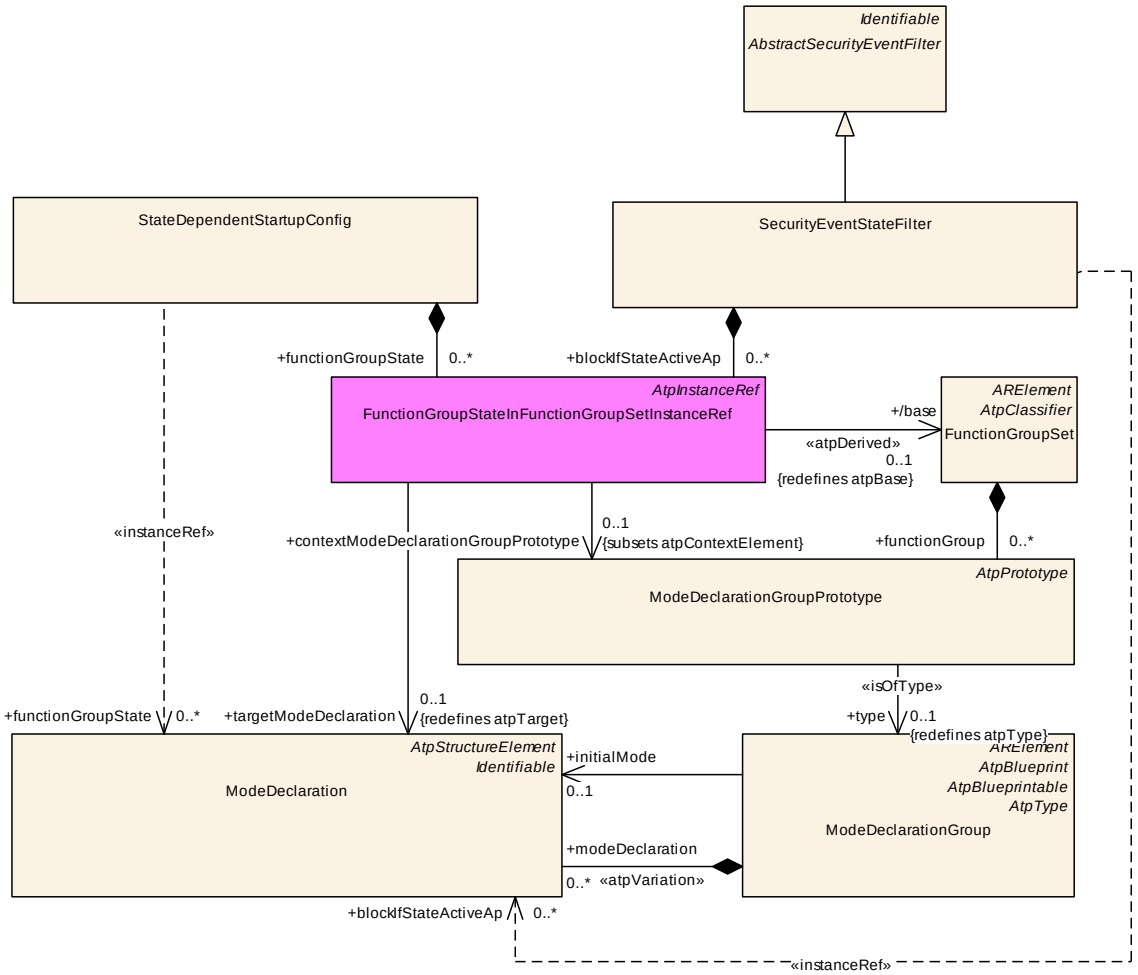


Figure C.14: Modeling of `FunctionGroupStateInFunctionGroupSetInstanceRef`

Class	**FunctionGroupStateInFunctionGroupSetInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest::InstanceRefs			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	FunctionGroupSet	0..1	ref	**Stereotypes:** atpDerived **Tags:**atp.Status=draft
contextMode Declaration GroupPrototype	ModeDeclarationGroup Prototype	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10
targetMode Declaration	ModeDeclaration	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20

Table C.14: FunctionGroupStateInFunctionGroupSetInstanceRef

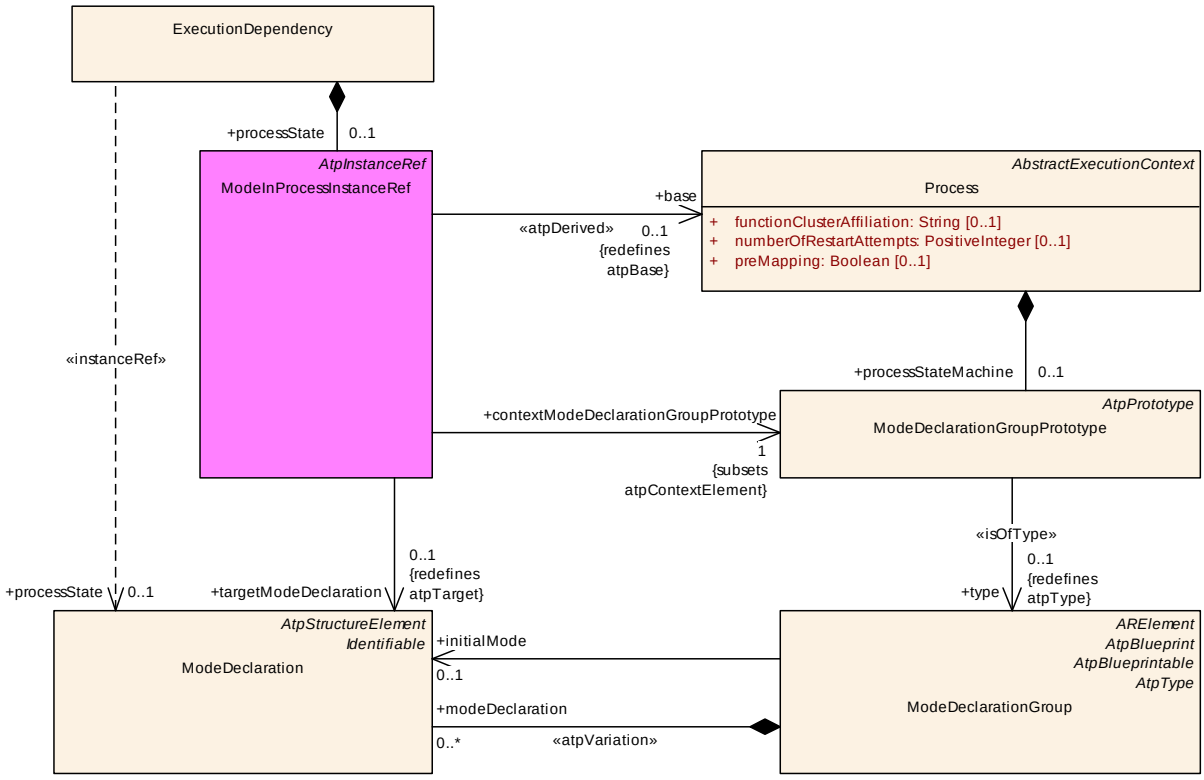


Figure C.15: Modeling of `ModeInProcessInstanceRef`

Class	**ModeInProcessInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest::InstanceRefs			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Process	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft xml.sequenceOffset=10
contextMode Declaration GroupPrototype	ModeDeclarationGroup Prototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetMode Declaration	ModeDeclaration	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30

Table C.15: ModeInProcessInstanceRef

C.6 Modeling of Diagnostic-related InstanceRefs

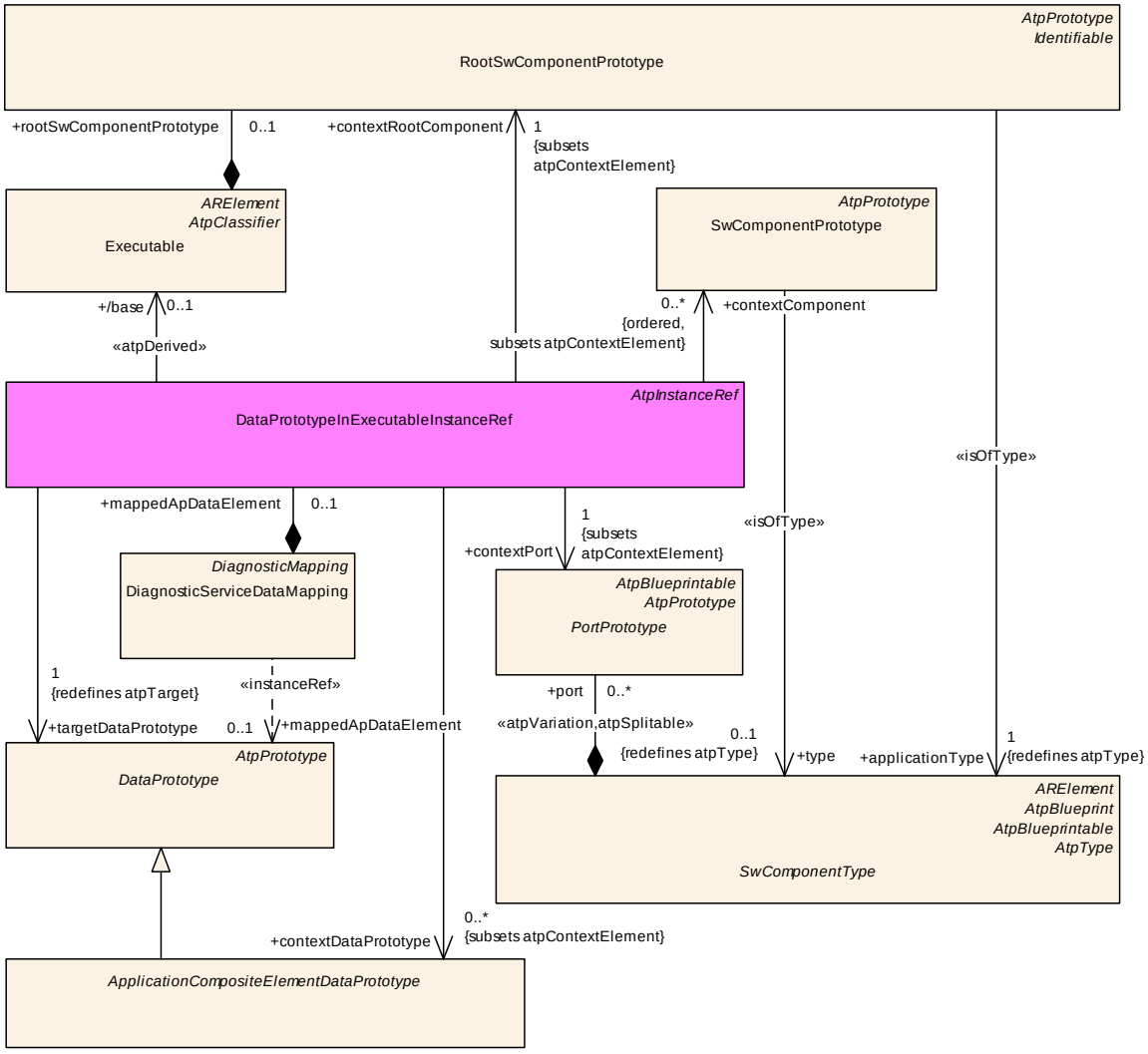


Figure C.16: Modeling of `DiagnosticServiceDataMapping` via `DataPrototypeInExecutableInstanceRef`

Class	**DataPrototypeInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft xml.sequenceOffset=10
context Component (ordered)	SwComponent Prototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30

△

Class	**DataPrototypeInExecutableInstanceRef**			
contextData Prototype	ApplicationComposite ElementDataPrototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=50
contextPort	PortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40
contextRoot Component	RootSwComponent Prototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetData Prototype	DataPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=60

Table C.16: DataPrototypeInExecutableInstanceRef

Class	**SwcServiceDependencyInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft xml.sequenceOffset=10
context Component (ordered)	SwComponent Prototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextRoot Component	RootSwComponent Prototype	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetSwc Service Dependency	SwcService Dependency	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40

Table C.17: SwcServiceDependencyInExecutableInstanceRef

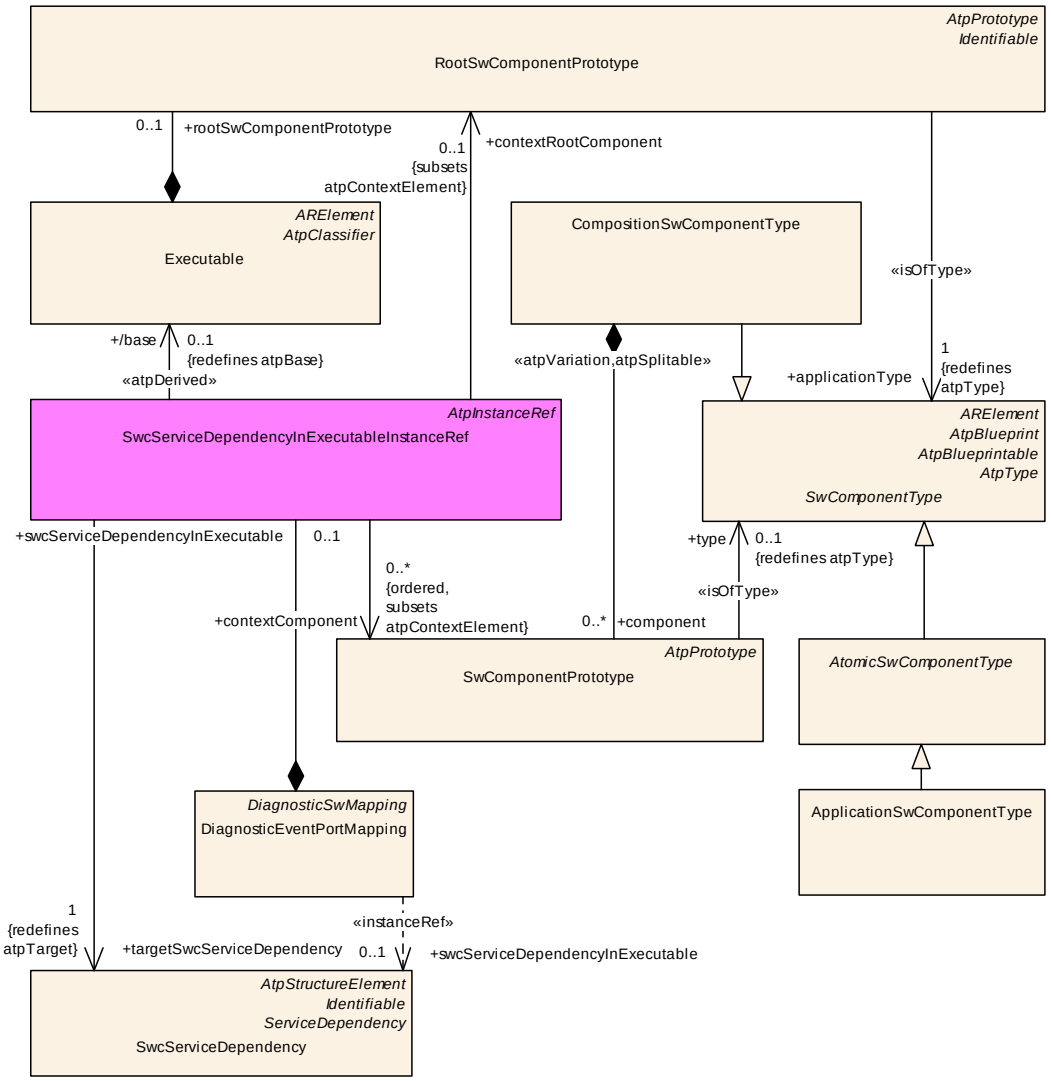


Figure C.17: Modeling of `DiagnosticEventPortMapping` via `SwcServiceDependencyInExecutableInstanceRef`

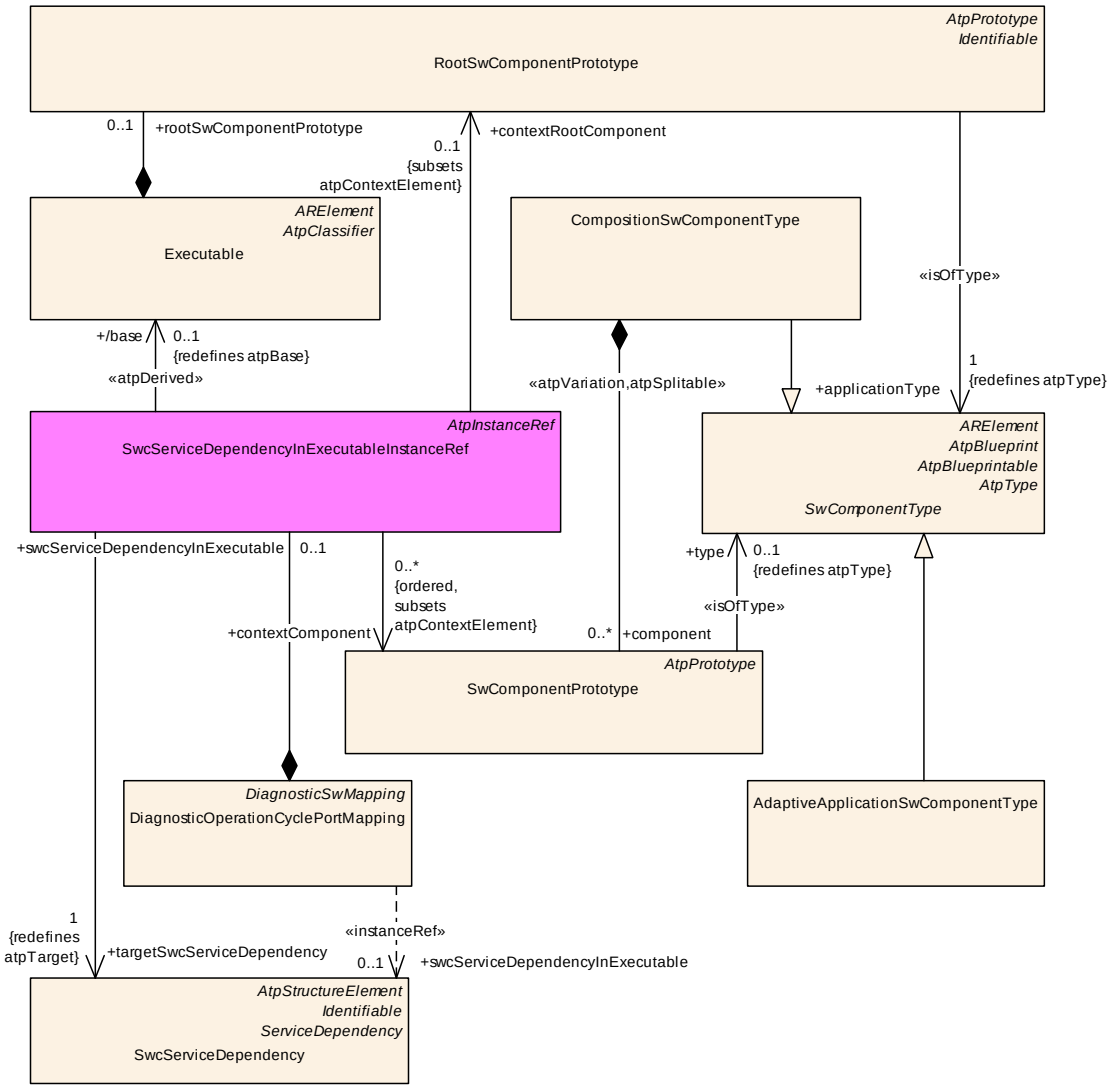


Figure C.18: Modeling of `DiagnosticOperationCyclePortMapping` via `SwcServiceDependencyInExecutableInstanceRef`

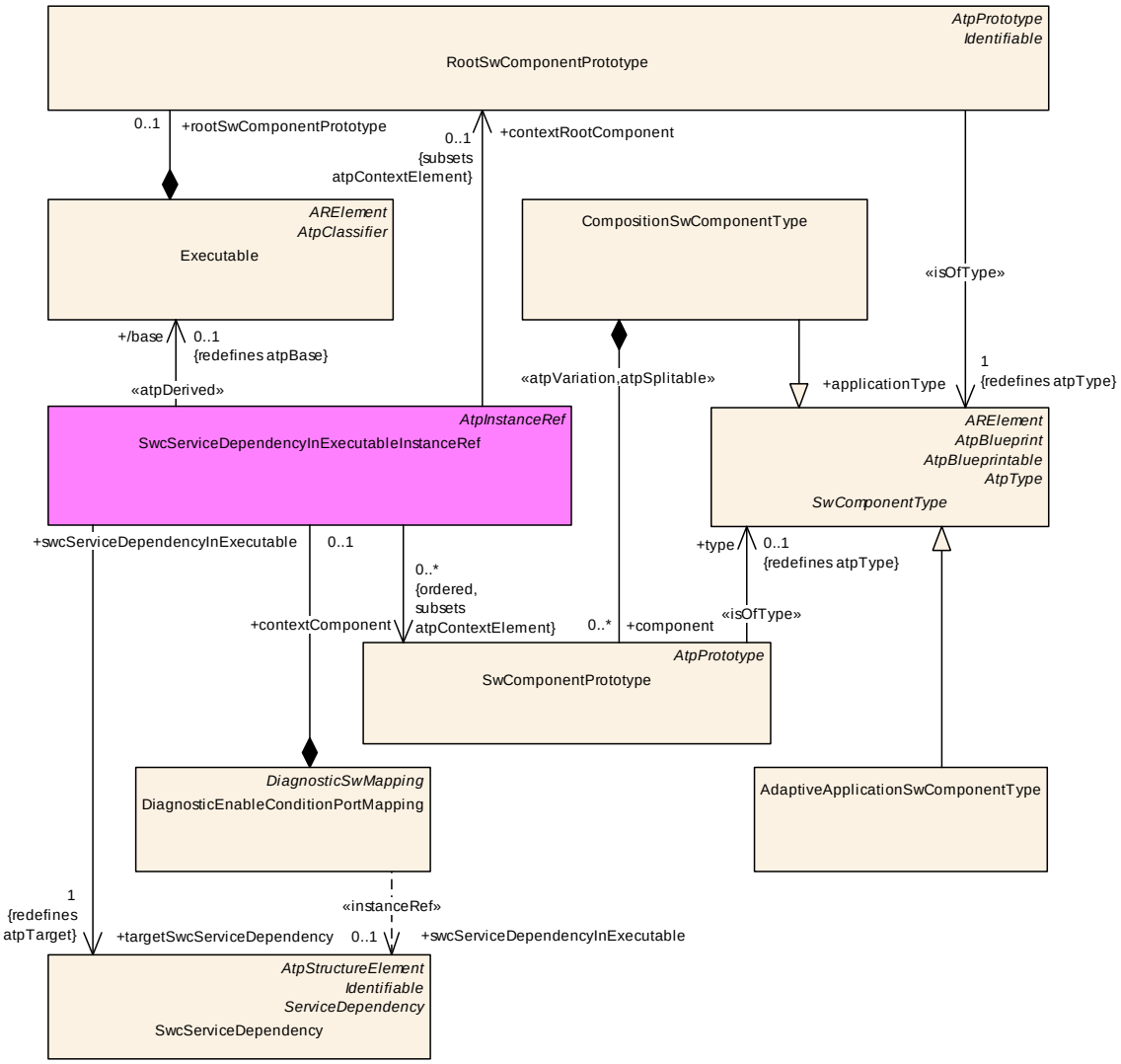


Figure C.19: Modeling of `DiagnosticEnableConditionPortMapping` via `SwcServiceDependencyInExecutableInstanceRef`

C.7 Modeling of REST-related InstanceRefs

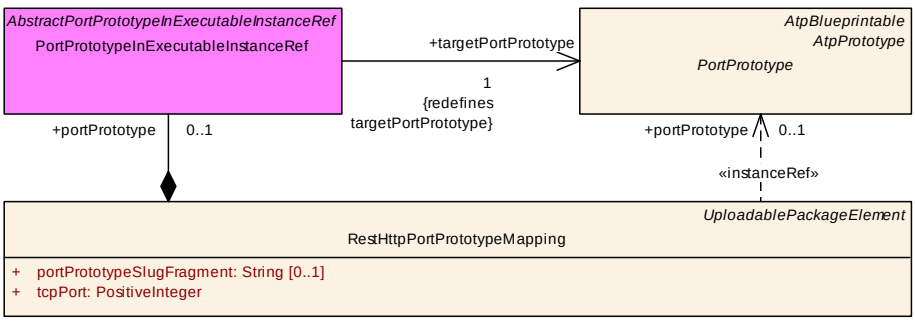


Figure C.20: Modeling of reference `RestHttpPortPrototypeMapping.portPrototype`

C.8 Modeling of PHM-related InstanceRefs

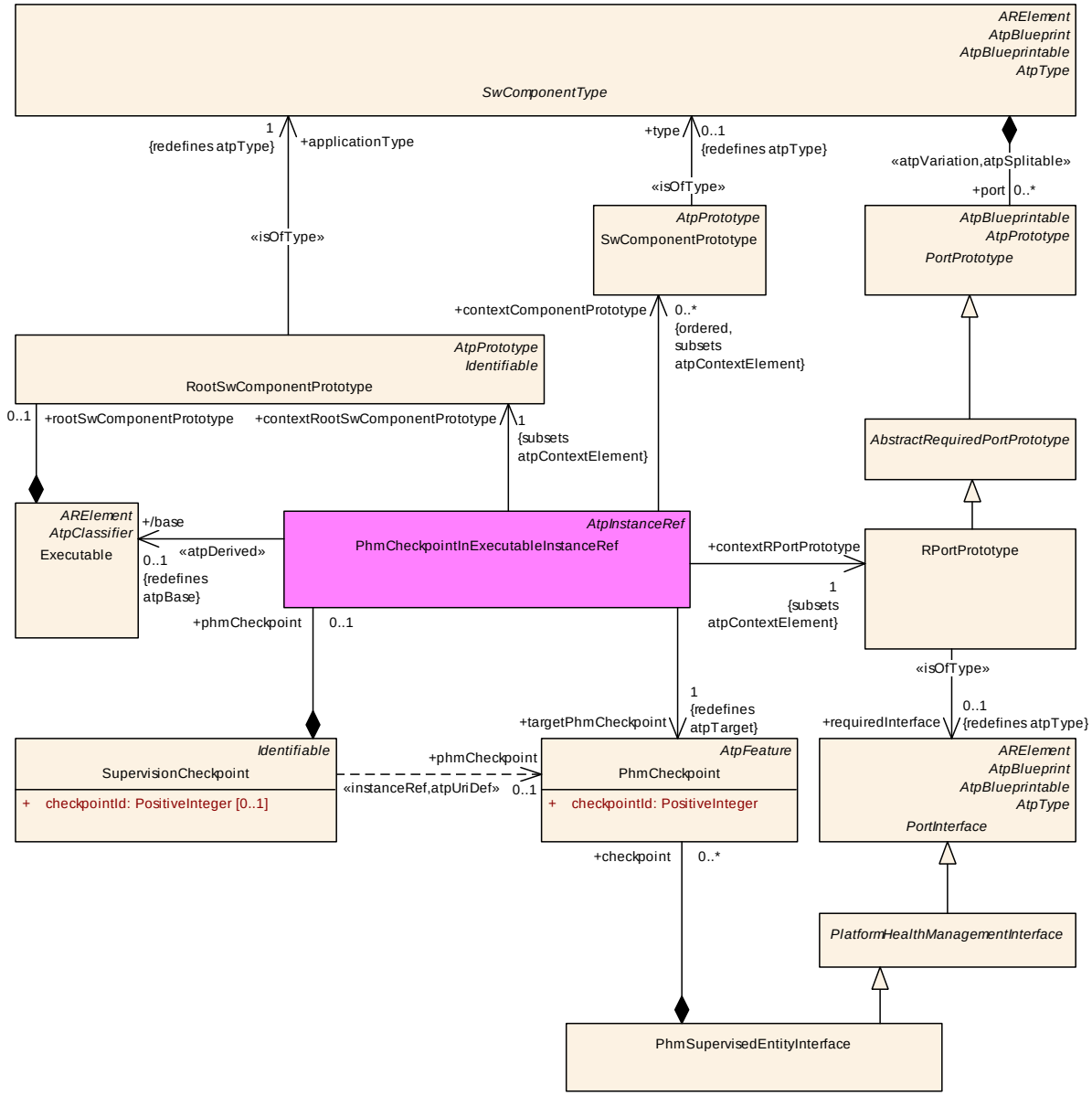


Figure C.21: Modeling of `PhmCheckpointInExecutableInstanceRef`

Class	**PhmCheckpointInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealth Management::InstanceRefs			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**PhmCheckpointInExecutableInstanceRef**			
base	Executable	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft xml.sequenceOffset=10
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
contextRPort Prototype	RPortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40
targetPhm Checkpoint	PhmCheckpoint	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=50

Table C.18: PhmCheckpointInExecutableInstanceRef

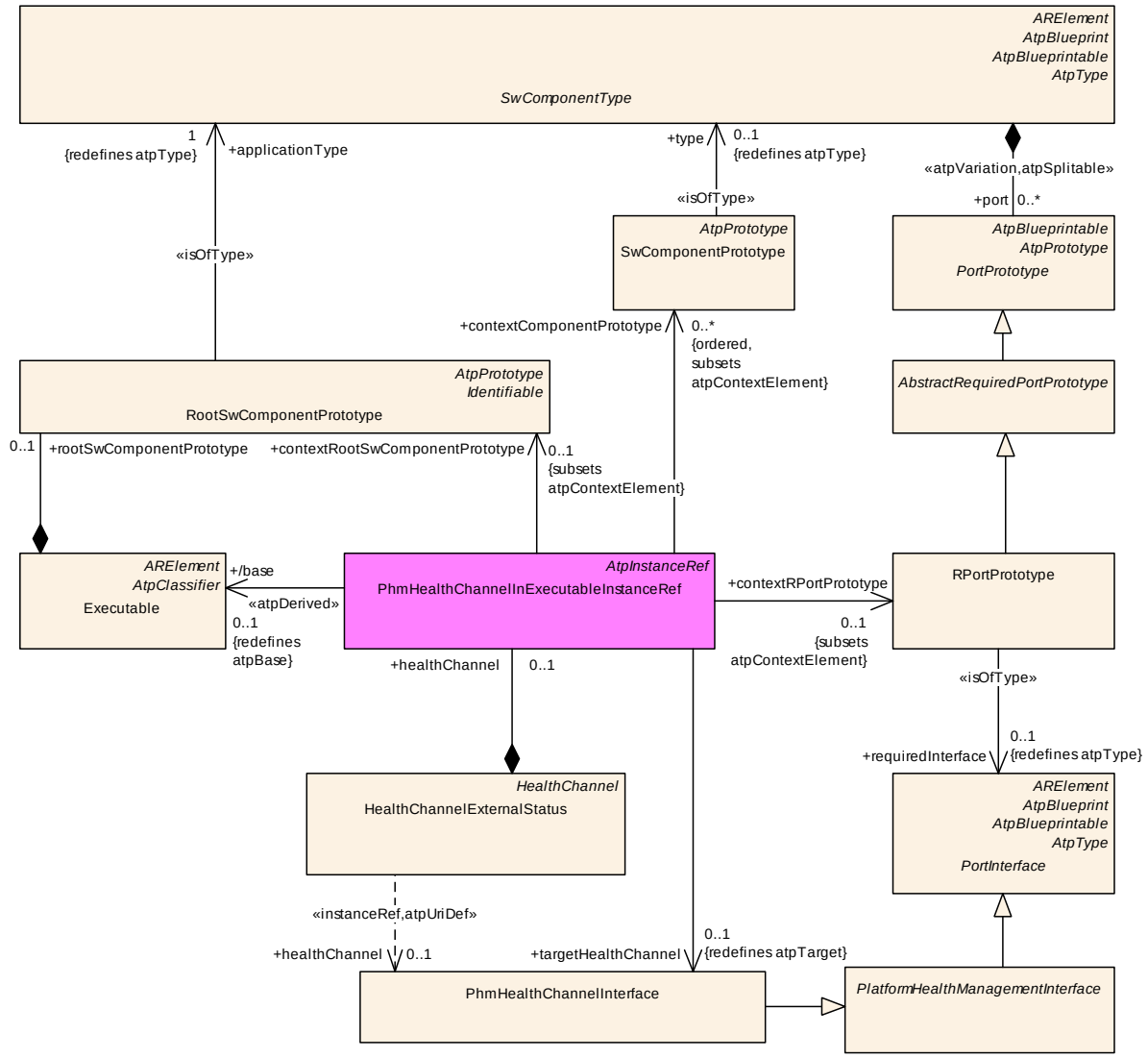


Figure C.22: Modeling of `PhmHealthChannelInExecutableInstanceRef`

Class	**PhmHealthChannelInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement::InstanceRefs			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft xml.sequenceOffset=10
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
contextRPort Prototype	RPortPrototype	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40
targetHealth Channel	PhmHealthChannel Interface	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=50

Table C.19: PhmHealthChannelInExecutableInstanceRef

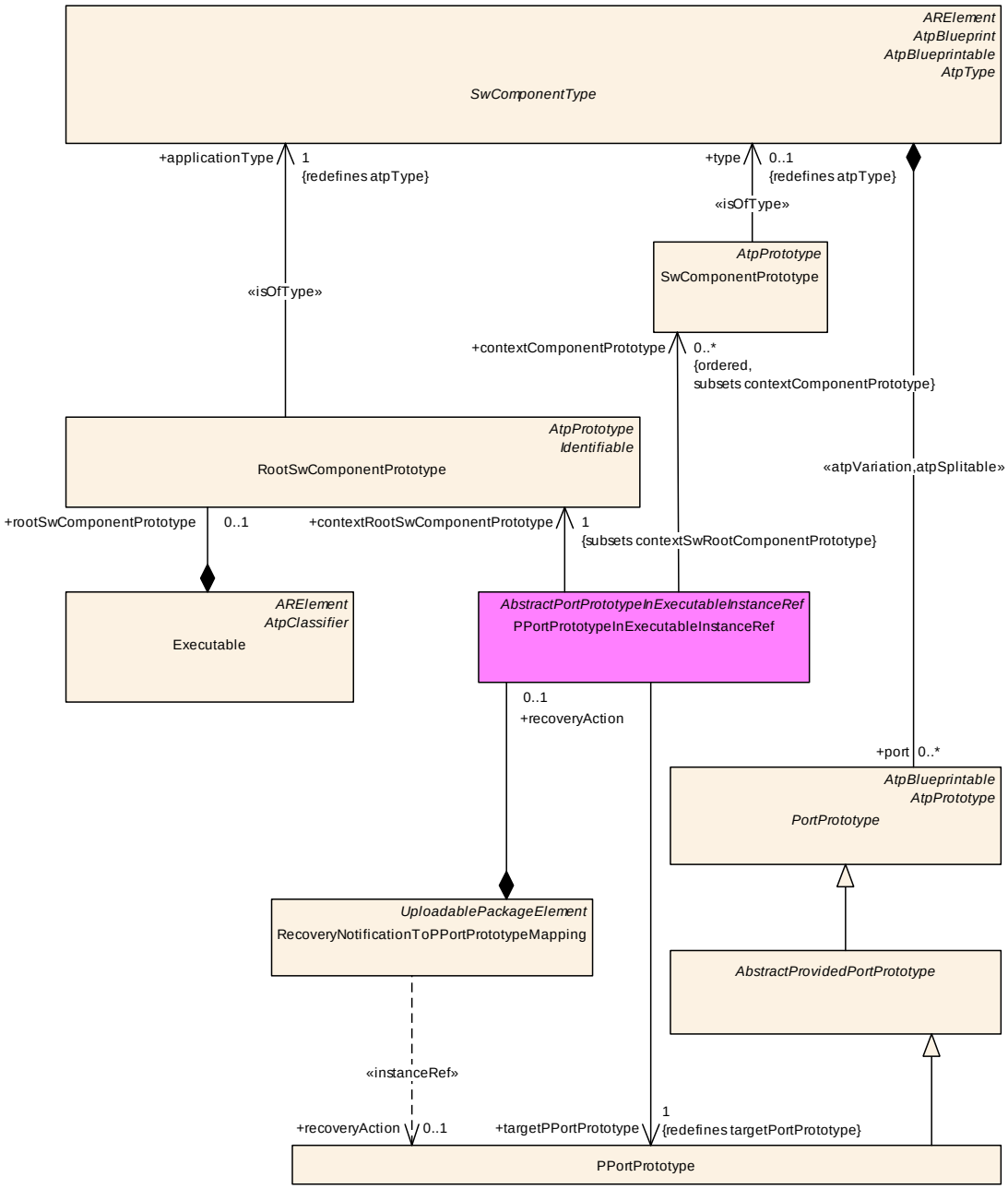


Figure C.23: Modeling of `RecoveryNotificationToPPortPrototypeMapping`

C.9 Modeling of Time-related InstanceRefs

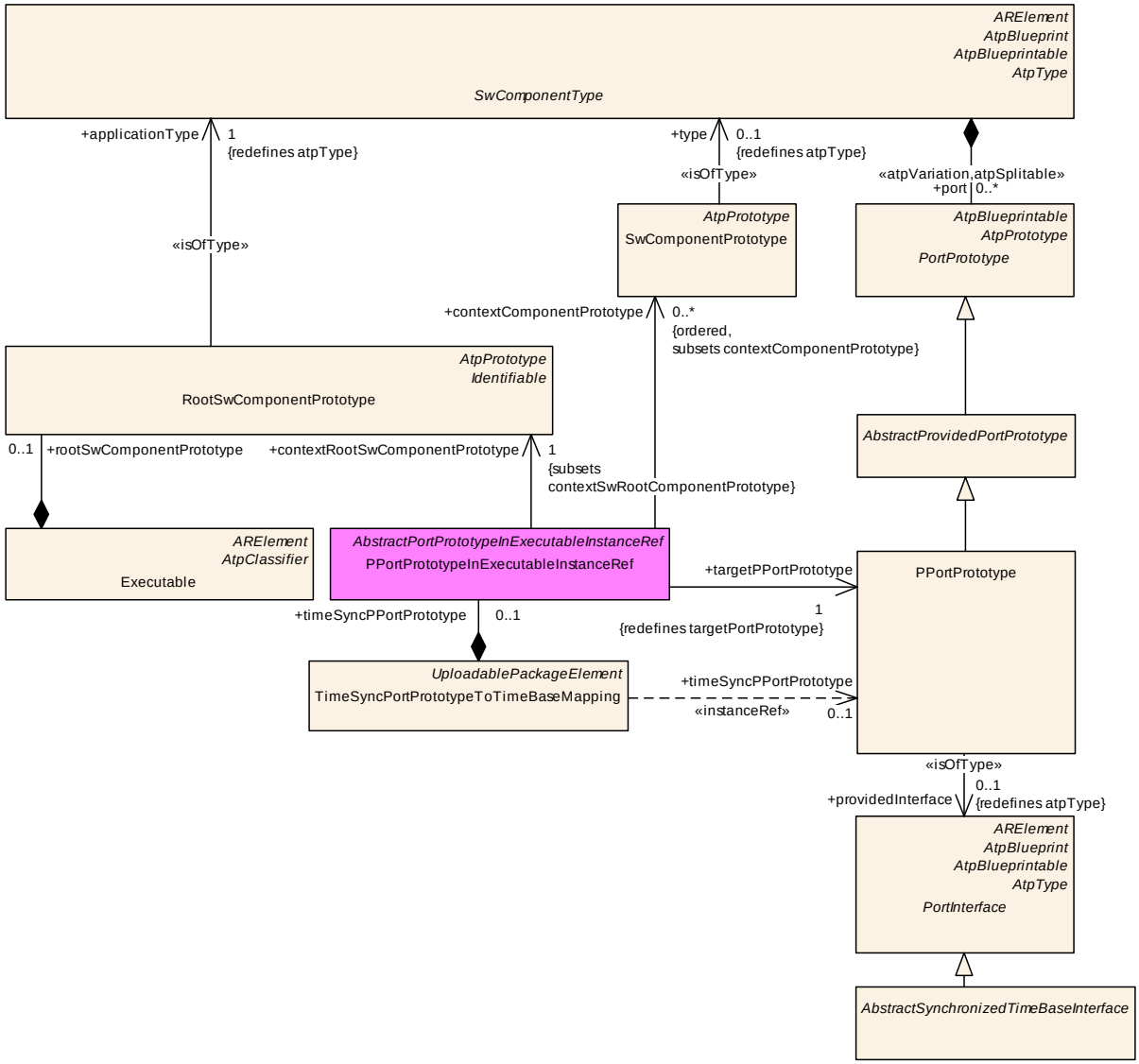


Figure C.24: Modeling of `TimeSyncPortPrototypeToTimeBaseMapping.timeSyncPPortPrototype`

C.10 Modeling of Persistency-related InstanceRefs

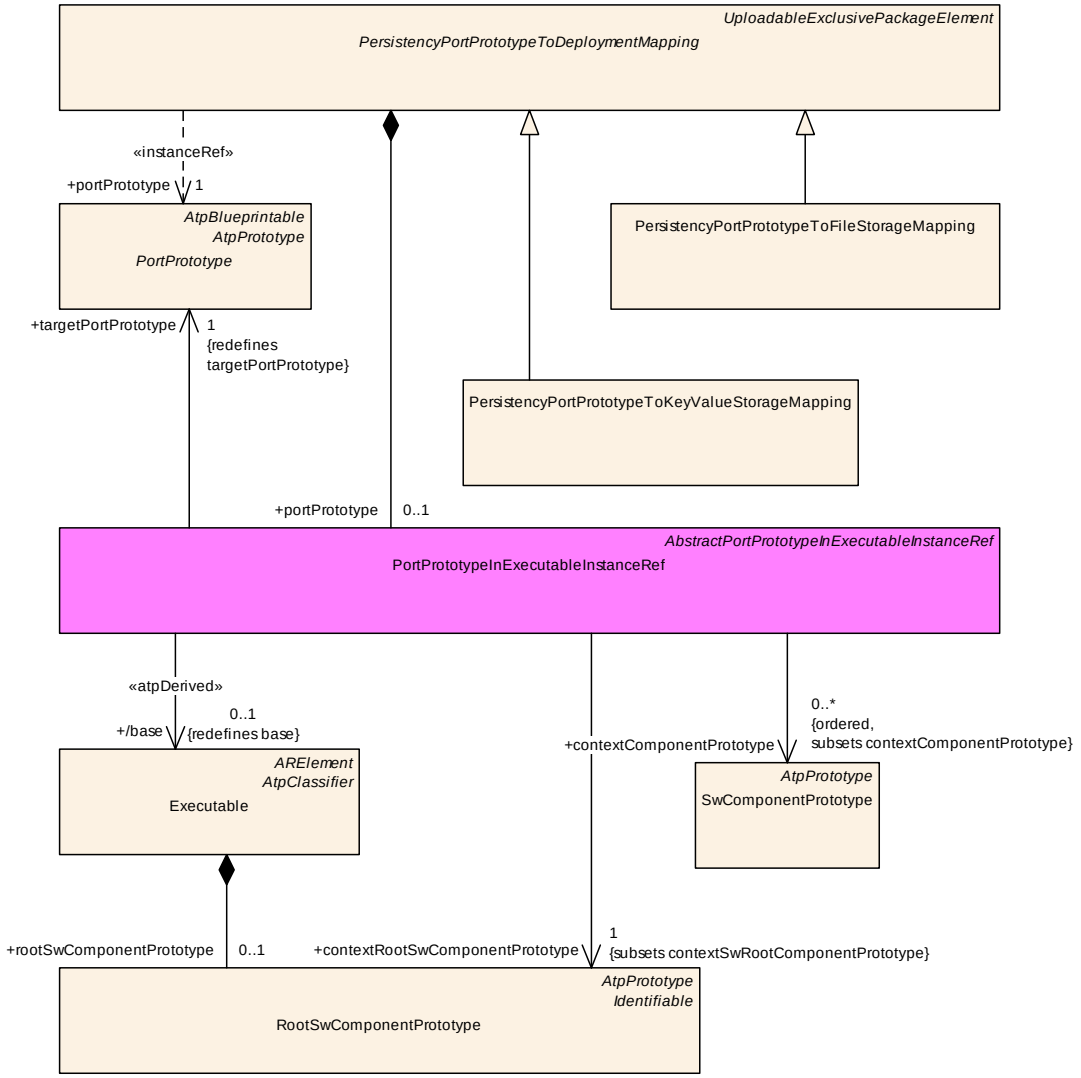


Figure C.25: Modeling of `PersistencyPortPrototypeToDeploymentMapping`

C.11 Modeling of diagnostic-related InstanceRefs

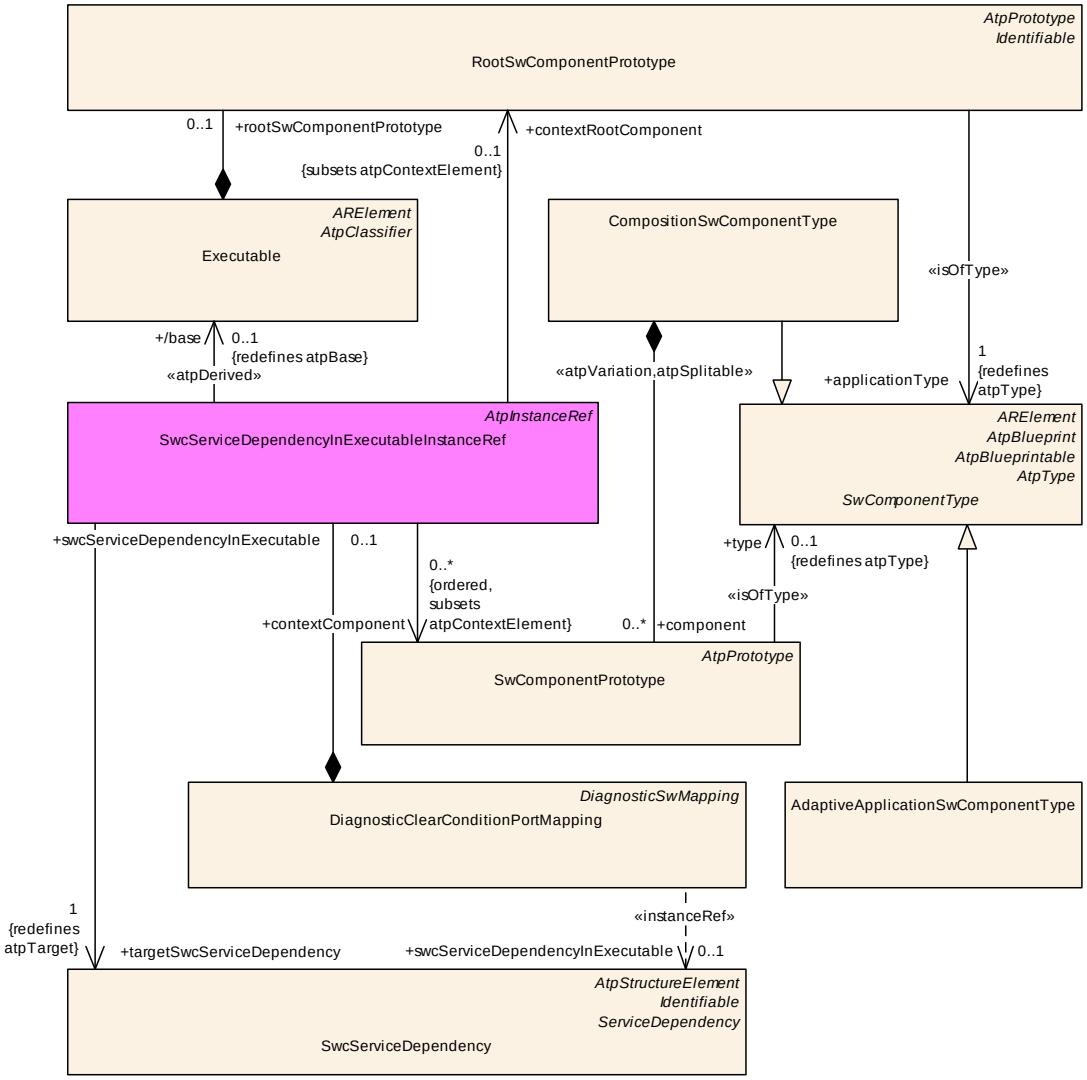


Figure C.26: Modeling of `DiagnosticClearConditionPortMapping`

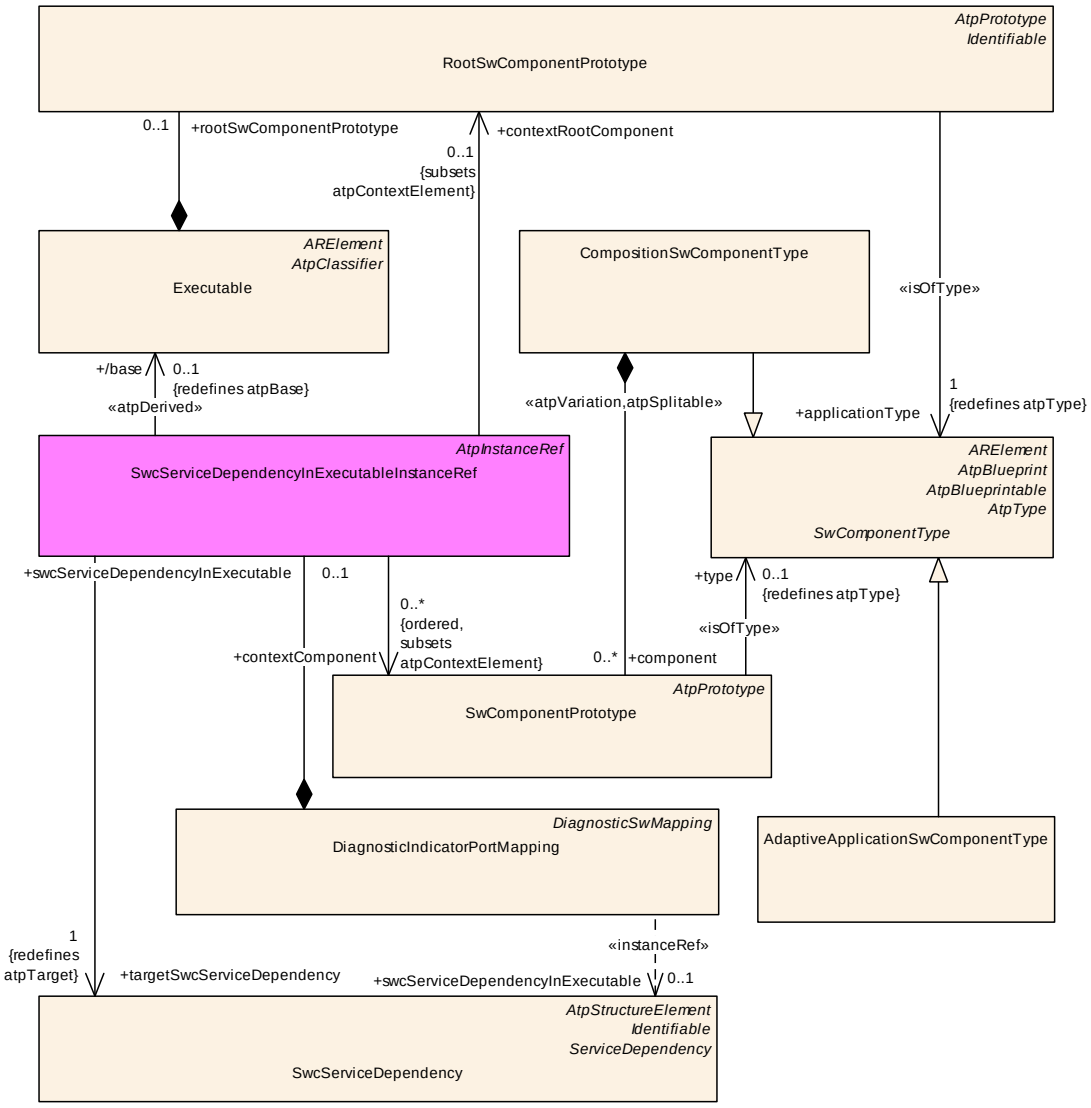


Figure C.27: Modeling of `DiagnosticIndicatorPortMapping`

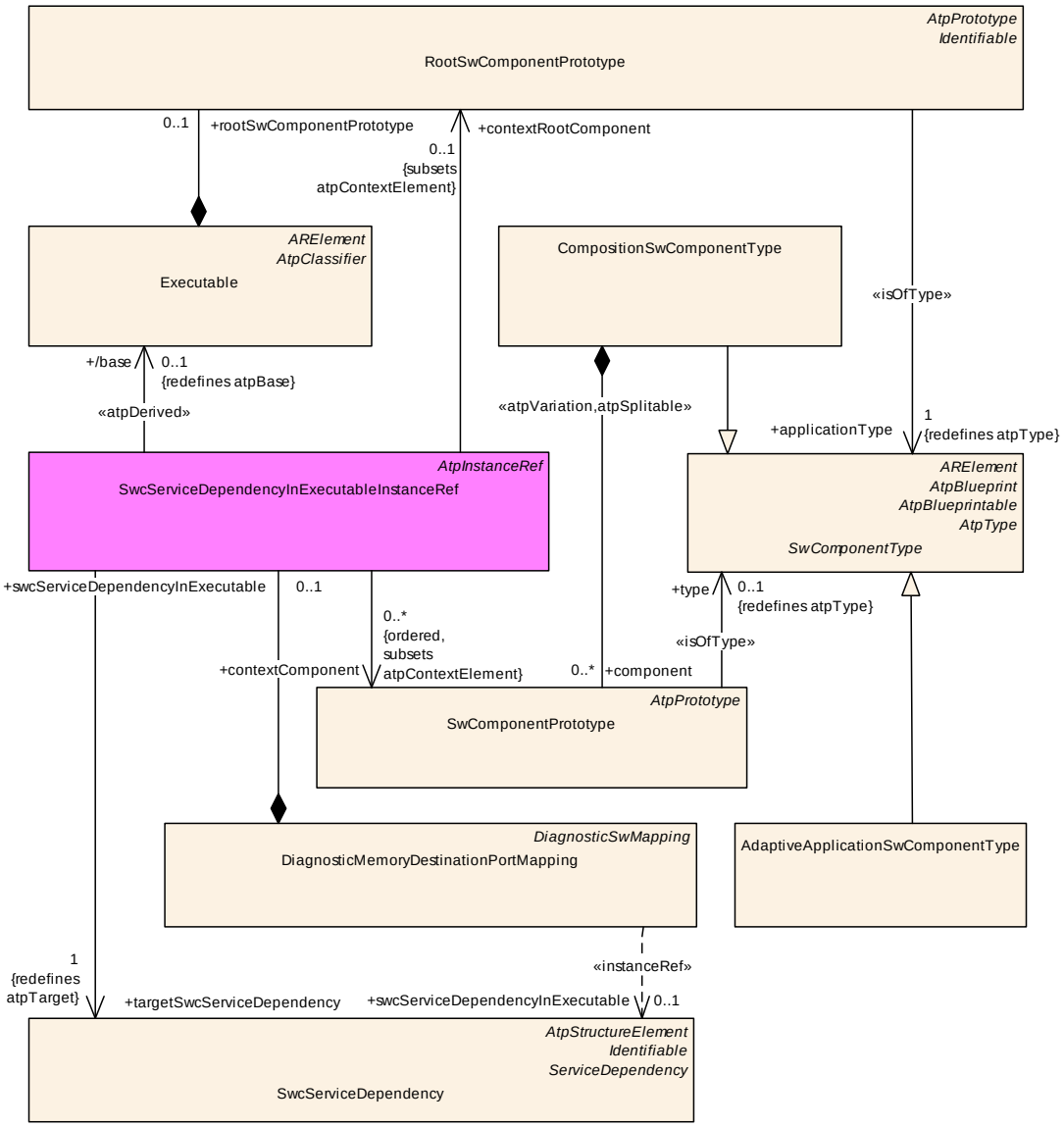


Figure C.28: Modeling of `DiagnosticMemoryDestinationPortMapping`

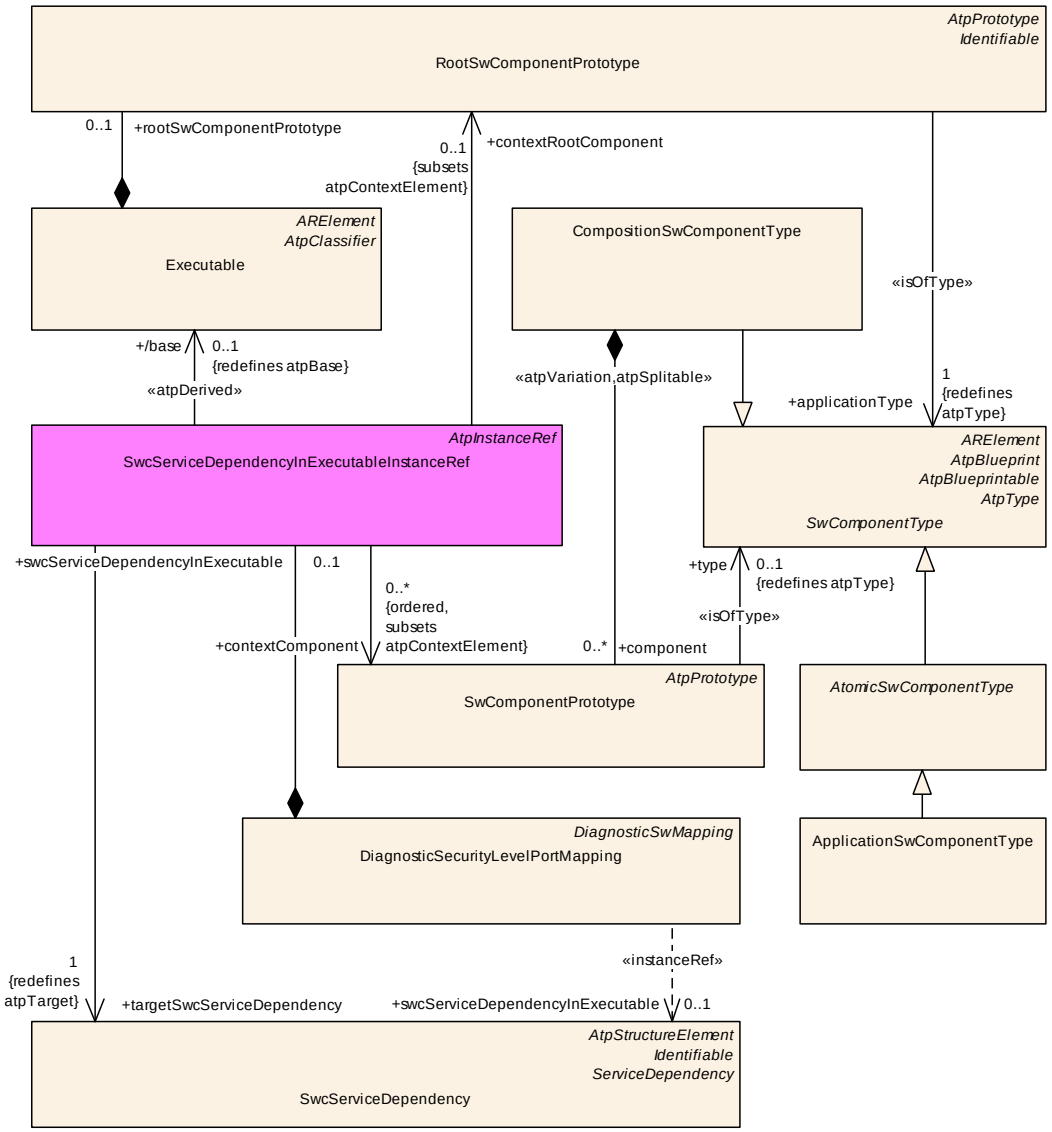


Figure C.29: Modeling of `DiagnosticSecurityLevelPortMapping`

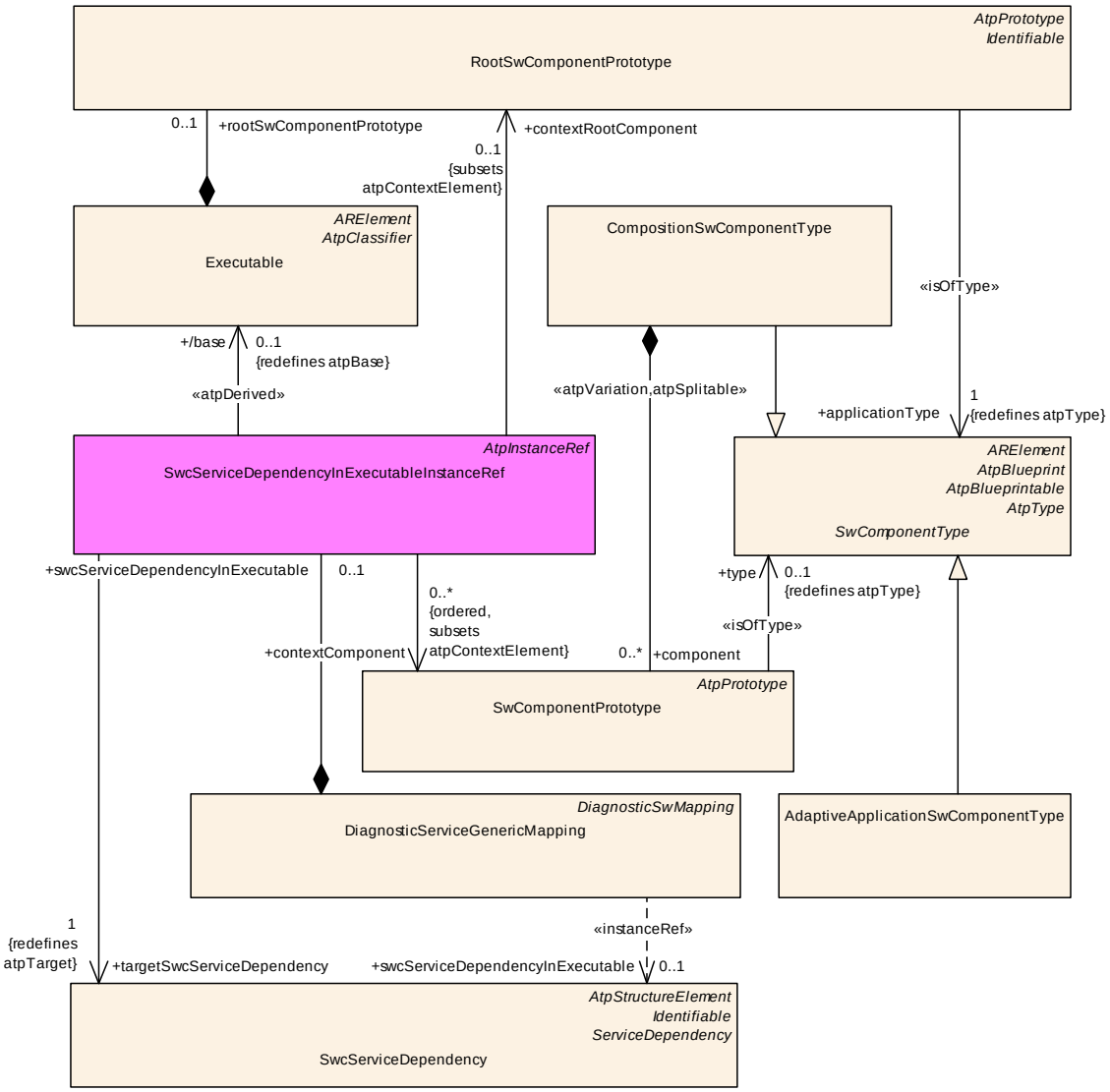


Figure C.30: Modeling of `DiagnosticServiceGenericMapping`

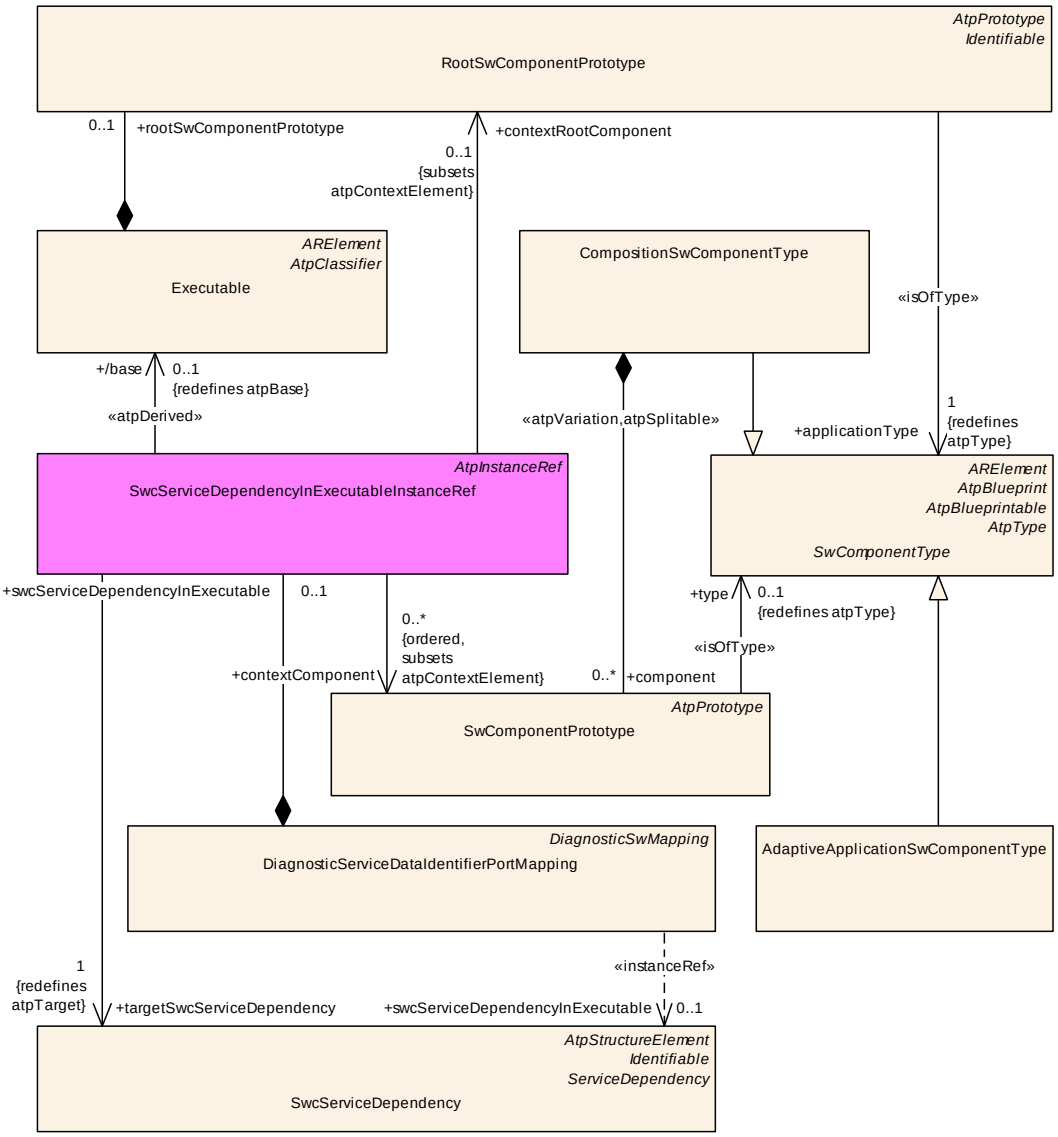


Figure C.31: Modeling of `DiagnosticServiceDataIdentifierPortMapping`

C.12 Modeling of SoftwareClusterDesign-related InstanceRefs

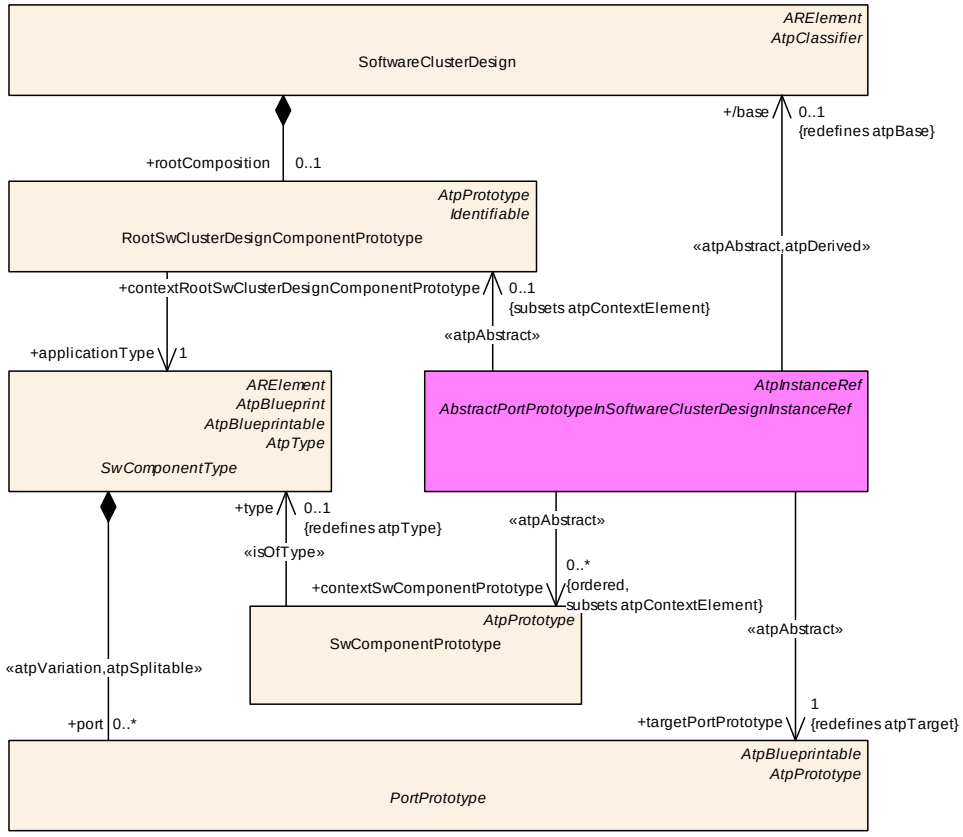


Figure C.32: Modeling of `AbstractPortPrototypeInSoftwareClusterDesignInstanceRef`

Class	***AbstractPortPrototypeInSoftwareClusterDesignInstanceRef*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInSoftwareClusterDesignInstanceRef			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Subclasses	PPortPrototypeInSoftwareClusterDesignInstanceRef, RPortPrototypeInSoftwareClusterDesignInstanceRef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	SoftwareClusterDesign	0..1	ref	**Stereotypes:** atpAbstract; atpDerived **Tags:**atp.Status=draft
contextRootSwClusterDesignComponentPrototype	RootSwClusterDesignComponentPrototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=10
contextSwComponentPrototype (ordered)	SwComponentPrototype	*	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=20

△

Class	***AbstractPortPrototypeInSoftwareClusterDesignInstanceRef*** (abstract)			
targetPort Prototype	PortPrototype	1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=30

Table C.20: AbstractPortPrototypeInSoftwareClusterDesignInstanceRef

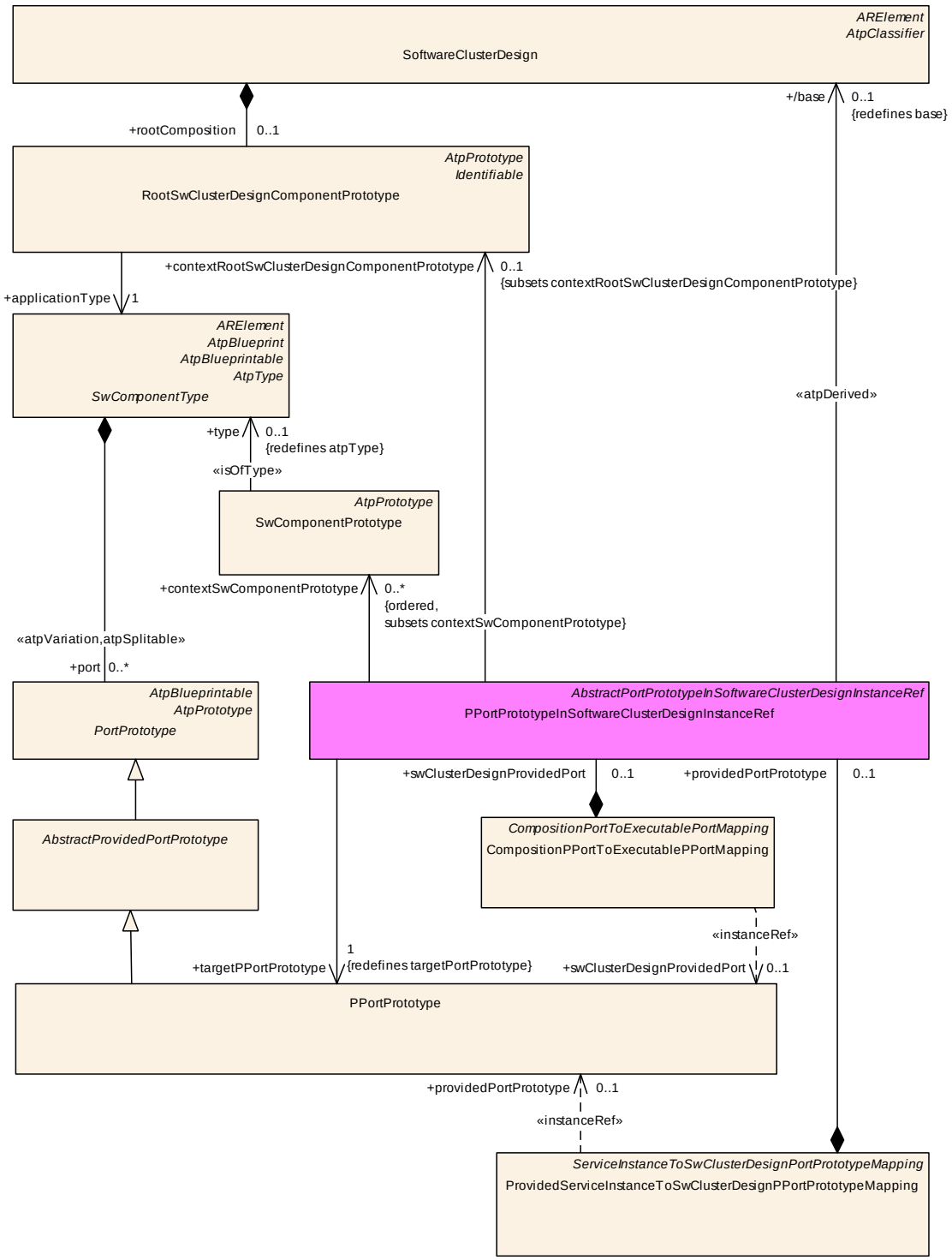


Figure C.33: Modeling of `PPortPrototypeInSoftwareClusterDesignInstanceRef`

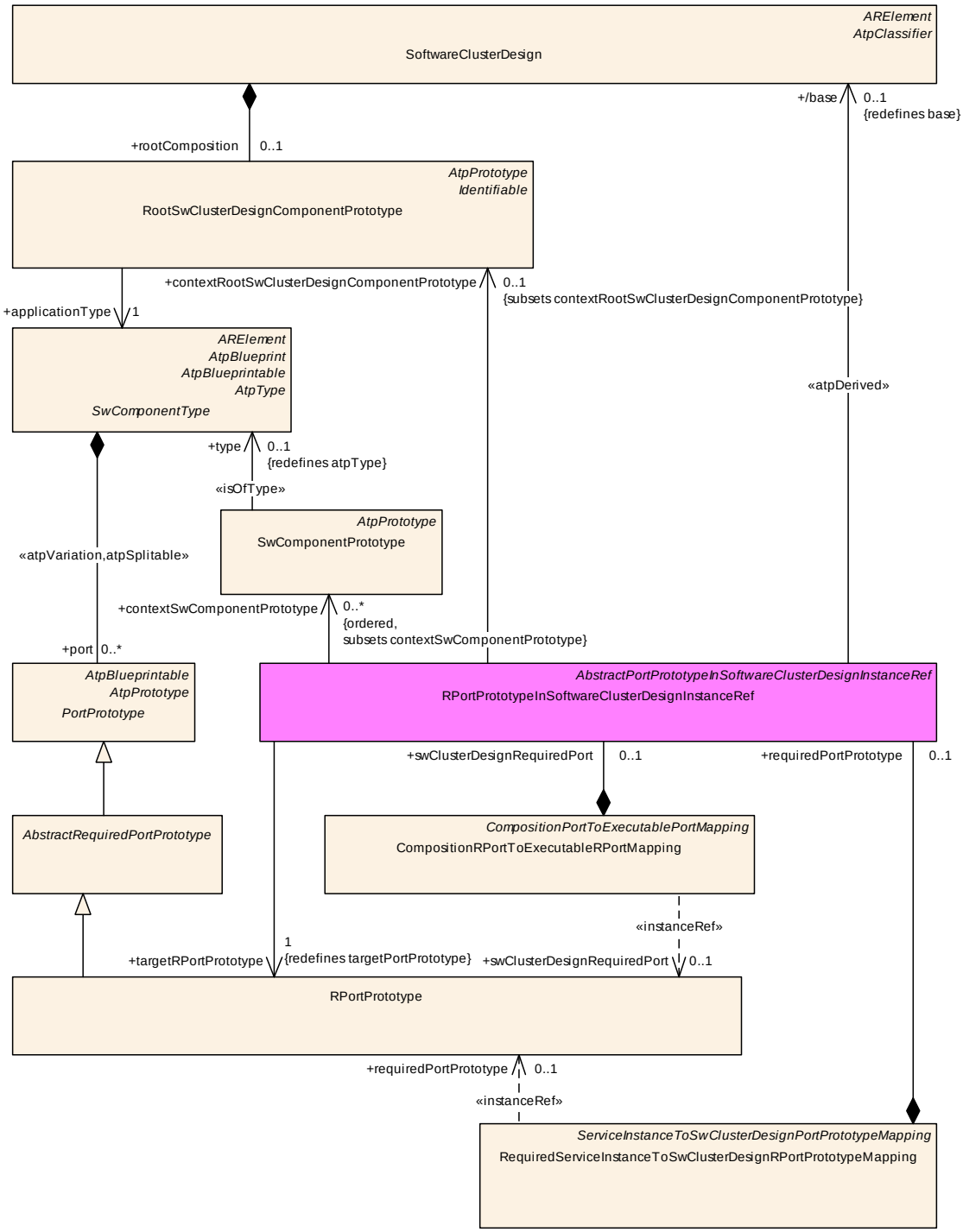


Figure C.34: Modeling of `RPortPrototypeInSoftwareClusterDesignInstanceRef`

D Mentioned Class Tables

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.

Class	***ARElement*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
Note	An element that can be defined stand-alone, i.e. without being part of another element (except for packages of course).			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractIamRemoteSubject*, AclObjectSet, AclOperation, AclPermission, AclRole, AliasNameSet, Allocator, ApApplicationError, ApApplicationErrorDomain, ApApplicationErrorSet, *AutosarDataType*, *BaseType*, BlueprintMappingSet, BuildActionManifest, CalibrationParameterValueSet, ClientIdDefinition Set, Collection, *CompositionPortToExecutablePortMapping*, CompuMethod, ConsistencyNeedsBlueprint Set, ConstantSpecification, ConstantSpecificationMappingSet, CryptoNeedToPortPrototypeMapping, CryptoServiceCertificate, CryptoServiceKey, CryptoServicePrimitive, CryptoServiceQueue, DataConstr, DataExchangePoint, DataTransformationSet, DataTypeMappingSet, *DiagnosticCommonElement*, DiagnosticConnection, DiagnosticContributionSet, DltLogChannelDesign, DltLogChannelDesignTo ProcessDesignMapping, Documentation, E2EProfileCompatibilityProps, E2EProfileConfigurationSet, EndToEndProtectionSet, EthIpProps, EthTcpIpIcmpProps, EthTcpIpProps, EvaluatedVariantSet, Executable, FMFeature, FMFeatureMap, FMFeatureModel, FMFeatureSelectionSet, FunctionGroupSet, GeneralPurposeConnection, *Grant*, *GrantDesign*, HwCategory, HwElement, HwType, IPSecConfig Props, *IdsCommonElement*, IdsDesign, InterfaceMappingSet, InterpolationRoutineMappingSet, Keyword Set, LifeCycleInfoSet, LifeCycleStateDefinitionGroup, Machine, McFunction, McGroup, ModeDeclaration Group, ModeDeclarationMappingSet, PhmContributionToMachineMapping, PhysicalDimension, Physical DimensionMappingSet, *PlatformModuleEndpointConfiguration*, *PortInterface*, PortInterfaceMappingSet, PortInterfaceToDataTypeMapping, PortPrototypeBlueprint, PostBuildVariantCriterion, PostBuildVariant CriterionValueSet, PredefinedVariant, ProcessDesign, ProcessDesignToMachineDesignMappingSet, RapidPrototypingScenario, SdgDef, SecureComPropsSet, SecurityEventReportToSecurityEvent DefinitionMapping, ServiceInstanceToSignalMappingSet, *ServiceInstanceToSwClusterDesignPort PrototypeMapping*, ServiceInterfaceMappingSet, ServiceInterfacePedigree, SignalServiceTranslation PropsSet, SoftwareCluster, SoftwareClusterDesign, SoftwarePackage, SomeipDataPrototype TransformationProps, SomeipSdClientEventGroupTimingConfig, SomeipSdClientServiceInstanceConfig, SomeipSdServerEventGroupTimingConfig, SomeipSdServerServiceInstanceConfig, SwAddrMethod, Sw AxisType, *SwComponentType*, SwRecordLayout, SwSystemconst, SwSystemconstantValueSet, System, SystemSignal, SystemSignalGroup, *TimingExtension*, TlvDataIdDefinitionSet, TransformationPropsSet, TransformationPropsToServiceInterfaceElementMappingSet, Unit, UnitGroup, *UploadablePackage Element*, VehiclePackage, ViewMapSet			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table D.1: ARElement

Class	**ARPackage**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
Note	AUTOSAR package, allowing to create top level packages to structure the contained ARElements. ARPackages are open sets. This means that in a file based description system multiple files can be used to partially describe the contents of a package. This is an extended version of MSR's SW-SYSTEM.			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ARPackage**			
arPackage	ARPackage	*	aggr	This represents a sub package within an ARPackage, thus allowing for an unlimited package hierarchy. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=arPackage.shortName, arPackage.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
element	PackageableElement	*	aggr	Elements that are part of this package **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=element.shortName, element.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=20
referenceBase	ReferenceBase	*	aggr	This denotes the reference bases for the package. This is the basis for all relative references within the package. The base needs to be selected according to the base attribute within the references. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=referenceBase.shortLabel xml.sequenceOffset=10

Table D.2: ARPackage

Class	***AbstractProvidedPortPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This abstract class provides the ability to become a provided PortPrototype.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Subclasses	PPortPrototype, PRPortPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
providedComSpec	PPortComSpec	*	aggr	Provided communication attributes per interface element (data element or operation).

Table D.3: AbstractProvidedPortPrototype

Class	***AbstractRequiredPortPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This abstract class provides the ability to become a required PortPrototype.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Subclasses	PRPortPrototype, RPortPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
requiredComSpec	RPortComSpec	*	aggr	Required communication attributes, one for each interface element.

Table D.4: AbstractRequiredPortPrototype

Class	**AbstractSignalBasedToISignalTriggeringMapping** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class is the common class for all SIgnalBased to ISignalTRiggering mappings. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	SignalBasedEventElementToISignalTriggeringMapping, SignalBasedFieldToISignalTriggeringMapping, SignalBasedMethodToISignalTriggeringMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table D.5: AbstractSignalBasedToISignalTriggeringMapping

Class	**AbstractSynchronizedTimeBaseInterface** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the abstract ability to define a PortInterface for the interaction with Time Synchronization. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	SynchronizedTimeBaseConsumerInterface, SynchronizedTimeBaseProviderInterface			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table D.6: AbstractSynchronizedTimeBaseInterface

Class	**AdminData**			
Package	M2::MSR::AsamHdo::AdminData			
Note	AdminData represents the ability to express administrative information for an element. This administration information is to be treated as meta-data such as revision id or state of the file. There are basically four kinds of meta-data • The language and/or used languages. • Revision information covering e.g. revision number, state, release date, changes. Note that this information can be given in general as well as related to a particular company. • Document meta-data specific for a company			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
docRevision (ordered)	DocRevision	*	aggr	This allows to denote information about the current revision of the object. Note that information about previous revisions can also be logged here. The entries shall be sorted descendant by date in order to reflect the history. Therefore the most recent entry representing the current version is denoted first. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=50 xml.typeElement=false xml.typeWrapperElement=false

▽

△

Class	**AdminData**			
language	LEnum	0..1	attr	This attribute specifies the master language of the document or the document fragment. The master language is the one in which the document is maintained and from which the other languages are derived from. In particular in case of inconsistencies, the information in the master language is priority. **Tags:**xml.sequenceOffset=20
sdg	Sdg	*	aggr	This property allows to keep special data which is not represented by the standard model. It can be utilized to keep e.g. tool specific data. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=sdg, sdg.variationPoint.shortLabel xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=60 xml.typeElement=false xml.typeWrapperElement=false
usedLanguages	MultiLanguagePlainText	0..1	aggr	This property specifies the languages which are provided in the document. Therefore it should only be specified in the top level admin data. For each language provided in the document there is one entry in MultilanguagePlain Text. The content of each entry can be used for illustration of the language. The used language itself depends on the language attribute in the entry. **Tags:**xml.sequenceOffset=30

Table D.7: AdminData

Class	**ApplicationArrayDataType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	An application data type which is an array, each element is of the same application data type. **Tags:**atp.recommendedPackage=ApplicationDataTypes			
Base	*ARElement*, *ARObject*, *ApplicationCompositeDataType*, *ApplicationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicArraySizeProfile	String	0..1	attr	Specifies the profile which the array will follow if it is a variable size array.
element	ApplicationArrayElement	0..1	aggr	This association implements the concept of an array element. That is, in some cases it is necessary to be able to identify single array elements, e.g. as input values for an interpolation routine.

Table D.8: ApplicationArrayDataType

Class	**ApplicationArrayElement**
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes
Note	Describes the properties of the elements of an application array data type.
Base	*ARObject*, *ApplicationCompositeElementDataPrototype*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	**ApplicationArrayElement**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arraySize Handling	ArraySizeHandling Enum	0..1	attr	The way how the size of the array is handled.
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls how the information about the array size shall be interpreted.
indexDataType	ApplicationPrimitive DataType	0..1	ref	This reference can be taken to assign a CompuMethod of category TEXTTABLE to the array. The texttable entries associate a textual value to an index number such that the element with that index number is represented by a symbolic name.
maxNumberOf Elements	PositiveInteger	0..1	attr	The maximum number of elements that the array can contain. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime

Table D.9: ApplicationArrayElement

Class	***ApplicationCompositeDataType*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	Abstract base class for all application data types composed of other data types.			
Base	*ARElement*, *ARObject*, *ApplicationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	ApplicationArrayDataType, ApplicationAssocMapDataType, ApplicationRecordDataType			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table D.10: ApplicationCompositeDataType

Class	**ApplicationError**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	This is a user-defined error that is associated with an element of an AUTOSAR interface. It is specific for the particular functionality or service provided by the AUTOSAR software component.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
errorCode	Integer	0..1	attr	The RTE generator is forced to assign this value to the corresponding error symbol. Note that for error codes certain ranges are predefined (see RTE specification).

Table D.11: ApplicationError

Class	**ApplicationRuleBasedValueSpecification**
Package	M2::AUTOSARTemplates::CommonStructure::Constants
Note	This meta-class represents rule based values for DataPrototypes typed by ApplicationDataTypes (ApplicationArrayDataType or a compound ApplicationPrimitiveDataType which also boils down to an array-nature).
Base	*ARObject*, *AbstractRuleBasedValueSpecification*, *ValueSpecification*

▽

△

Class	**ApplicationRuleBasedValueSpecification**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
category	Identifier	0..1	attr	This represents the category of the RuleBasedValue Specification **Tags:**xml.sequenceOffset=-20
swAxisCont (ordered)	RuleBasedAxisCont	*	aggr	This represents the axis values of a Compound Primitive Data Type (curve or map). The first swAxisCont describes the x-axis, the second sw AxisCont describes the y-axis, the third swAxisCont describes the z-axis. In addition to this, the axis can be denoted in swAxisIndex.
swValueCont	RuleBasedValueCont	0..1	aggr	This represents the values of an array or Compound Primitive Data Type.

Table D.12: ApplicationRuleBasedValueSpecification

Class	**ApplicationSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	The ApplicationSwComponentType is used to represent the application software. **Tags:**atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table D.13: ApplicationSwComponentType

Class	**ApplicationValueSpecification**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This meta-class represents values for DataPrototypes typed by ApplicationDataTypes (this includes in particular compound primitives). For further details refer to ASAM CDF 2.0. This meta-class corresponds to some extent with SW-INSTANCE in ASAM CDF 2.0.			
Base	*ARObject*, *ValueSpecification*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
category	Identifier	0..1	attr	Specifies to which category of ApplicationDataType this ApplicationValueSpecification can be applied (e.g. as an initial value), thus imposing constraints on the structure and semantics of the contained values, see [constr_1006] and [constr_2051].
swAxisCont (ordered)	SwAxisCont	*	aggr	This represents the axis values of a Compound Primitive Data Type (curve or map). The first swAxisCont describes the x-axis, the second sw AxisCont describes the y-axis, the third swAxisCont describes the z-axis. In addition to this, the axis can be denoted in swAxisIndex.
swValueCont	SwValueCont	0..1	aggr	This represents the values of a Compound Primitive Data Type.

Table D.14: ApplicationValueSpecification

Class	**ArgumentDataPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	An argument of an operation, much like a data element, but also carries direction information and is owned by a particular ClientServerOperation.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
direction	ArgumentDirection Enum	0..1	attr	This attribute specifies the direction of the argument prototype.
serverArgument ImplPolicy	ServerArgumentImpl PolicyEnum	0..1	attr	This defines how the argument type of the servers RunnableEntity is implemented. If the attribute is not defined this has the same semantics as if the attribute is set to the value useArgumentType for primitive arguments and structures.

Table D.15: ArgumentDataPrototype

Enumeration	**ArgumentDirectionEnum**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	Use cases: • Arguments in ClientServerOperation can have different directions that need to be formally indicated because they have an impact on how the function signature looks like eventually. • Arguments in BswModuleEntry already determine a function signature, but the direction is used to specify the semantics, especially of pointer arguments.
Literal	***Description***
in	The argument value is passed to the callee. **Tags:**atp.EnumerationLiteralIndex=0
inout	The argument value is passed to the callee but also passed back from the callee to the caller. **Tags:**atp.EnumerationLiteralIndex=1
out	The argument value is passed from the callee to the caller. **Tags:**atp.EnumerationLiteralIndex=2

Table D.16: ArgumentDirectionEnum

Enumeration	**ArraySizeSemanticsEnum**
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes
Note	This type controls how the information about the number of elements in an ApplicationArrayDataType is to be interpreted.
Literal	***Description***
fixedSize	This means that the ApplicationArrayDataType will always have a fixed number of elements. **Tags:**atp.EnumerationLiteralIndex=0
variableSize	This implies that the actual number of elements in the ApplicationArrayDataType might vary at run-time. The value of arraySize represents the maximum number of elements in the array. **Tags:**atp.EnumerationLiteralIndex=1

Table D.17: ArraySizeSemanticsEnum

Class	**ArrayValueSpecification**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Specifies the values for an array.			
Base	*ARObject*, *CompositeValueSpecification*, *ValueSpecification*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
element (ordered)	ValueSpecification	*	aggr	The value for a single array element. All Value Specifications aggregated by ArrayValueSpecification shall have the same structure. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
intendedPartial Initialization Count	PositiveInteger	0..1	attr	This attribute shall only have a meaning for dynamic arrays and shall be taken as a sanity check: the number filled in the attribute shall be identical to the number of ArrayValueSpecification.element. If the attribute does not exist it means that no partial initialization is intended.

Table D.18: ArrayValueSpecification

Class	**AssemblySwConnector**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	AssemblySwConnectors are exclusively used to connect SwComponentPrototypes in the context of a CompositionSwComponentType.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SwConnector*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
provider	AbstractProvidedPort Prototype	0..1	iref	Instance of providing port. **InstanceRef implemented by:**PPortInComposition InstanceRef
requester	AbstractRequiredPort Prototype	0..1	iref	Instance of requiring port. **InstanceRef implemented by:**RPortInComposition InstanceRef

Table D.19: AssemblySwConnector

Class	***AutosarDataPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Base class for prototypical roles of an AutosarDataType.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ArgumentDataPrototype, Field, ParameterDataPrototype, PersistencyDataElement, VariableData Prototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
type	AutosarDataType	0..1	tref	This represents the corresponding data type. **Stereotypes:** isOfType

Table D.20: AutosarDataPrototype

Class	***AutosarDataType*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	Abstract base class for user defined AUTOSAR data types for software.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractImplementationDataType*, *ApplicationDataType*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
swDataDefProps	SwDataDefProps	0..1	aggr	The properties of this AutosarDataType.

Table D.21: AutosarDataType

Class	**BaseTypeDirectDefinition**			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This BaseType is defined directly (as opposite to a derived BaseType)			
Base	*ARObject*, *BaseTypeDefinition*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
baseTypeEncoding	BaseTypeEncodingString	0..1	attr	This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence. **Tags:**xml.sequenceOffset=90
baseTypeSize	PositiveInteger	0..1	attr	Describes the length of the data type specified in the container in bits. **Tags:**xml.sequenceOffset=70
byteOrder	ByteOrderEnum	0..1	attr	This attribute specifies the byte order of the base type. **Tags:**xml.sequenceOffset=110
memAlignment	PositiveInteger	0..1	attr	This attribute describes the alignment of the memory object in bits. E.g. "8" specifies, that the object in question is aligned to a byte while "32" specifies that it is aligned four byte. If the value is set to "0" the meaning shall be interpreted as "unspecified". **Tags:**xml.sequenceOffset=100
nativeDeclaration	NativeDeclarationString	0..1	attr	This attribute describes the declaration of such a base type in the native programming language, primarily in the Programming language C. This can then be used by a code generator to include the necessary declarations into a header file. For example BaseType with shortName: "MyUnsignedInt" native Declaration: "unsigned short" Results in typedef unsigned short MyUnsignedInt; If the attribute is not defined the referring Implementation DataTypes will not be generated as a typedef by RTE. If a nativeDeclaration type is given it shall fulfill the characteristic given by basetypeEncoding and baseTypeSize. This is required to ensure the consistent handling and interpretation by software components, RTE, COM and MCM systems. **Tags:**xml.sequenceOffset=120

Table D.22: BaseTypeDirectDefinition

Class	**ClientServerInterface**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A client/server interface declares a number of operations that can be invoked on a server by a client. **Tags:**atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
possibleError	ApplicationError	*	aggr	Application errors that are defined as part of this interface.

Table D.23: ClientServerInterface

Class	***ComGrant*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM			
Note	This meta-class serves as the abstract base class for defining specific ComGrants **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	ComEventGrant, ComFieldGrant, ComMethodGrant			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
remoteSubject	AbstractIamRemoteSubject	*	ref	This optional reference defines the remoteSubject that is allowed to access the defined Object via the Grant. **Tags:**atp.Status=draft
serviceInstance	AdaptivePlatformServiceInstance	1	ref	This reference identifies the applicable AdaptivePlatform ServiceInstance for which the grant applies. **Tags:**atp.Status=draft

Table D.24: ComGrant

Class	***ComGrantDesign*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class serves as an abstract base class for the description of com grants on design level. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	ComEventGrantDesign, ComFieldGrantDesign, ComMethodGrantDesign			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
remoteSubject	AbstractIamRemoteSubject	0..1	ref	This optional reference defines the remoteSubject that is allowed to access the defined Object via the Grant. **Tags:**atp.Status=draft

Table D.25: ComGrantDesign

Class	***CommConnectorPort*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	The Ecu communication relationship defines which signals, Pdus and frames are actually received and transmitted by this ECU. For each signal, Pdu or Frame that is transmitted or received and used by the Ecu an association between an ISignalPort, IPduPort or FramePort with the corresponding Triggering shall be created. An ISignalPort shall be created only if the corresponding signal is handled by COM (RTE or Signal Gateway). If a Pdu Gateway ECU only routes the Pdu without being interested in the content only a FramePort and an IPduPort needs to be created.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	FramePort, IPduPort, ISignalPort			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Direction	Communication DirectionType	1	attr	Communication Direction of the Connector Port (input or output Port).

Table D.26: CommConnectorPort

Enumeration	**CommunicationDirectionType**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication
Note	Describes the communication direction.
Literal	***Description***
in	Reception (Input) **Tags:**atp.EnumerationLiteralIndex=0
out	Transmission (Output) **Tags:**atp.EnumerationLiteralIndex=1

Table D.27: CommunicationDirectionType

Class	**CompuConst**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the fact that the value of a computation method scale is constant.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compuConst ContentType	CompuConstContent	0..1	aggr	This is the actual content of the constant compu method scale. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=10 xml.typeElement=false xml.typeWrapperElement=false

Table D.28: CompuConst

Class	**CompuConstTextContent**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the textual content of a scale.			
Base	*ARObject*, *CompuConstContent*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
vt	VerbatimString	0..1	attr	This represents a textual constant in the computation method.

Table D.29: CompuConstTextContent

Class	**CompuMethod**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the ability to express the relationship between a physical value and the mathematical representation. Note that this is still independent of the technical implementation in data types. It only specifies the formula how the internal value corresponds to its physical pendant. **Tags:**atp.recommendedPackage=CompuMethods			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compuInternalToPhys	Compu	0..1	aggr	This specifies the computation from internal values to physical values. **Tags:**xml.sequenceOffset=80
compuPhysToInternal	Compu	0..1	aggr	This represents the computation from physical values to the internal values. **Tags:**xml.sequenceOffset=90
displayFormat	DisplayFormatString	0..1	attr	This property specifies, how the physical value shall be displayed e.g. in documents or measurement and calibration tools. **Tags:**xml.sequenceOffset=20
unit	Unit	0..1	ref	This is the physical unit of the Physical values for which the CompuMethod applies. **Tags:**xml.sequenceOffset=30

Table D.30: CompuMethod

Class	**CompuScale**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the ability to specify one segment of a segmented computation method.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compuInverseValue	CompuConst	0..1	aggr	This is the inverse value of the constraint. This supports the case that the scale is not reversible per se. **Tags:**xml.sequenceOffset=60

▽

△

Class	**CompuScale**			
compuScale Contents	CompuScaleContents	0..1	aggr	This represents the computation details of the scale. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=70 xml.typeElement=false xml.typeWrapperElement=false
desc	MultiLanguageOverview Paragraph	0..1	aggr	<desc> represents a general but brief description of the object in question. **Tags:**xml.sequenceOffset=30
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
mask	PositiveInteger	0..1	attr	In difference to all the other computational methods every COMPU-SCALE will be applied including the bit MASK. Therefore it is allowed for this type of COMPU-METHOD, that COMPU-SCALES overlap. To calculate the string reverse to a value, the string has to be split and the according value for each substring has to be summed up. The sum is finally transmitted. The processing has to be done in order of the COMPU-SCALE elements. **Tags:**xml.sequenceOffset=35
shortLabel	Identifier	0..1	attr	This element specifies a short name for the particular scale. The name can for example be used to derive a programming language identifier. **Tags:**xml.sequenceOffset=20
symbol	CIdentifier	0..1	attr	The symbol, if provided, is used by code generators to get a C identifier for the CompuScale. The name will be used as is for the code generation, therefore it needs to be unique within the generation context. **Tags:**xml.sequenceOffset=25
upperLimit	Limit	0..1	attr	This specifies the upper limit of a of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=50

Table D.31: CompuScale

Class	**ConstantReference**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Instead of defining this value inline, a constant is referenced.			
Base	*ARObject*, *ValueSpecification*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
constant	ConstantSpecification	0..1	ref	The referenced constant.

Table D.32: ConstantReference

Class	**CouplingPort**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	A CouplingPort is used to connect a CouplingElement with an EcuInstance or two CouplingElements with each other via a CouplingPortConnection. Optionally, the CouplingPort may also have a reference to a macMulticastGroup and a defaultVLAN.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
connection Negotiation Behavior	EthernetConnection NegotiationEnum	0..1	attr	Specifies the connection negotiation of the CouplingPort. **Tags:**atp.Status=shallBecomeMandatory
couplingPort Details	CouplingPortDetails	0..1	aggr	Defines more details of a CouplingPort in case a more specific configuration is required.
couplingPort Role	CouplingPortRoleEnum	0..1	attr	Defines the role this CouplingPort takes in the context of the CouplingElement.
defaultVlan	EthernetPhysical Channel	0..1	ref	The vLanIdentifier of the referenced VLAN is the Default-PVID (port VLAN ID). A Port VLAN ID is a default VLAN ID that is assigned to an access CouplingPort to designate the VLAN segment to which this port is connected. Also, if a CouplingPort has not been configured with any VLAN memberships, the virtual switch's Port VLAN ID (pvid) becomes the default VLAN ID for the ports connection. This identifier/tag is added for incoming untagged messages at the port (ingress tagging). For outgoing messages with this identifier, the tag is removed at the port (egress untagging, depending on the Vlan Membership.sendActivity).
macLayerType	EthernetMacLayerType Enum	0..1	attr	Specifies the mac layer type of the CouplingPort.
macMulticast Address	MacMulticastGroup	*	ref	Assigns a set of MAC-Multicast-Addresses which are addressable via this CouplingPort. This is a static pre-configuration and further addresses may be learned during runtime.
physicalLayer Type	EthernetPhysicalLayer TypeEnum	0..1	attr	Specifies the physical layer type of the CouplingPort.
plcaProps	PlcaProps	0..1	aggr	Optional properties for configuration of PLCA (Physical Layer Collision Avoidance) in case 10-BASE-T1S Ethernet is used and PLCA is enabled on the Coupling Port (PHY). **Tags:**atp.Status=draft
pncMapping	PncMappingIdent	*	ref	Reference to the partial networks this CouplingPort participates in.
receiveActivity	EthernetSwitchVlan IngressTagEnum	0..1	attr	Defines the handling of frames at the ingress port.
vlan Membership	VlanMembership	*	aggr	Messages of VLANs that are defined here can be communicated via the CouplingPort.
vlanModifier	EthernetPhysical Channel	0..1	ref	All incoming messages at this CouplingPort shall be tagged with this VLAN Id. This tagging is performed regardless whether the message already has a VLAN tag or is untagged, an existing VLAN tag will be overwritten. This feature is XOR with CoupligPort.defaultVlan.
wakeupSleep OnDataline Config	EthernetWakeupSleep OnDatalineConfig	0..1	ref	Optional reference to EthernetWakeupSleepOnDataline Config. **Tags:**atp.Status=draft

Table D.33: CouplingPort

Class	**CouplingPortConnection**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Connection between two CouplingPorts (firstPort and secondPort) or between a collection of Ports that are all referenced by the portCollection reference.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
firstPort	CouplingPort	0..1	ref	Reference to the first CouplingPort that is connected via the CouplingPortConnection.
nodePort	CouplingPort	*	ref	Reference to a number of CouplingPorts that are connected via the CouplingPortConnection. This reference shall be used to describe a 10BASE-T1S topology architecture where several CouplingPorts of EthernetCommunicationControllers are connected via one CouplingPortConnection. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=nodePort.couplingPort, nodePort.variation Point.shortLabel atp.Status=draft vh.latestBindingTime=postBuild
plcaLocalNode Count	PositiveInteger	0..1	attr	Defines the number of communication participants in case 10BASE-T1S and the nodePort reference is used. **Tags:**atp.Status=draft
plcaTransmit Opportunity Timer	PositiveInteger	0..1	attr	Timer for the transmission in bit time to evaluate if a Transmission Opportunity is yield or not. **Tags:**atp.Status=draft
secondPort	CouplingPort	0..1	ref	Reference to the second CouplingPort that is connected via the CouplingPortConnection.

Table D.34: CouplingPortConnection

Class	**CryptoCertificateToPortPrototypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to define a mapping between a CryptoCertificate on deployment level to a given PortPrototype that is typed by a CryptoCertificateInterface. **Tags:** atp.Status=draft atp.recommendedPackage=CryptoCertificateToPortPrototypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
crypto Certificate	CryptoCertificate	1	ref	This reference represents the mapped cryptoCertificate. **Tags:**atp.Status=draft
portPrototype	PortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **Tags:**atp.Status=draft **InstanceRef implemented by:**PortPrototypeIn ExecutableInstanceRef
process	Process	1	ref	This reference represents the process required as context for the mapping. **Tags:**atp.Status=draft

▽

△

Class	CryptoCertificateToPortPrototypeMapping			
writeAccess	Boolean	0..1	attr	This attribute defines whether the application has write-access to the CryptoCertificate (True) or only read-access (False).

Table D.35: CryptoCertificateToPortPrototypeMapping

Class	CryptoKeySlotToPortPrototypeMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to define a mapping between a CryptoKeySlot on deployment level to a given PortPrototype that is typed by a CryptoKeySlotInterface. **Tags:** atp.Status=draft atp.recommendedPackage=CryptoKeySlotToPortPrototypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
keySlot	CryptoKeySlot	1	ref	This reference represents the mapped CryptoKeySlot. **Tags:**atp.Status=draft
portPrototype	PortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **Tags:**atp.Status=draft **InstanceRef implemented by:**PortPrototypeIn ExecutableInstanceRef
process	Process	1	ref	This reference represents the process required as context for the mapping. **Tags:**atp.Status=draft

Table D.36: CryptoKeySlotToPortPrototypeMapping

Class	CryptoProviderToPortPrototypeMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to define a mapping between a CryptoProvider on deployment level to a given PortPrototype that is typed by a CryptoProviderInterface. **Tags:** atp.Status=draft atp.recommendedPackage=CryptoProviderToPortPrototypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
cryptoProvider	CryptoProvider	1	ref	This reference represents the mapped cryptoProvider. **Tags:**atp.Status=draft
portPrototype	PortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **Tags:**atp.Status=draft **InstanceRef implemented by:**PortPrototypeIn ExecutableInstanceRef
process	Process	1	ref	This reference represents the process required as context for the mapping. **Tags:**atp.Status=draft

Table D.37: CryptoProviderToPortPrototypeMapping

Class	CyclicTiming			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication::Timing			
Note	Specification of a cyclic sending behavior.			
Base	*ARObject*, *Describable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
timeOffset	TimeRangeType	0..1	aggr	This attribute specifies the time until first transmission of this I-PDU. This attribute defines the time between Com_IpduGroupStart and the first transmission of the cyclic part of this transmission request for this I-PDU.
timePeriod	TimeRangeType	1	aggr	Period of the repetition of cyclic transmissions.

Table D.38: CyclicTiming

Class	DataConstr			
Package	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
Note	This meta-class represents the ability to specify constraints on data. **Tags:**atp.recommendedPackage=DataConstrs			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataConstrRule	DataConstrRule	*	aggr	This is one particular rule within the data constraints. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

Table D.39: DataConstr

Class	DataConstrRule			
Package	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
Note	This meta-class represents the ability to express one specific data constraint rule.			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
constrLevel	Integer	0..1	attr	This attribute describes the category of a constraint. One of its functions is in the area of constraint violation, where it can be used from a certain level, to produce error messages. The lower the level, the more stringent the check. Used to distinguish hard or soft limits. **Tags:**xml.sequenceOffset=20
internalConstrs	InternalConstrs	0..1	aggr	Describes the limitations applicable on the internal domain (as opposed to the physical domain). **Tags:**xml.sequenceOffset=40
physConstrs	PhysConstrs	0..1	aggr	Describes the limitations applicable on the physical domain (as opposed to the internal domain). **Tags:**xml.sequenceOffset=30

Table D.40: DataConstrRule

Class	**DataFilter**			
Package	M2::AUTOSARTemplates::CommonStructure::Filter			
Note	Base class for data filters. The type of the filter is specified in attribute dataFilterType. Some of the filter types require additional arguments which are specified as attributes of this class.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataFilterType	DataFilterTypeEnum	0..1	attr	This attribute specifies the type of the filter.
mask	UnlimitedInteger	0..1	attr	Mask for old and new value.
max	UnlimitedInteger	0..1	attr	Value to specify the upper boundary
min	UnlimitedInteger	0..1	attr	Value to specify the lower boundary
offset	PositiveInteger	0..1	attr	Specifies the initial number of messages to occur before the first message is passed
period	PositiveInteger	0..1	attr	Specifies number of messages to occur before the message is passed again
x	UnlimitedInteger	0..1	attr	Value to compare with

Table D.41: DataFilter

Enumeration	**DataFilterTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Filter
Note	This enum specifies the supported DataFilterTypes.
Literal	***Description***
always	No filtering is performed so that the message always passes. **Tags:**atp.EnumerationLiteralIndex=0
maskedNewDiffers MaskedOld	Pass messages where the masked value has changed. (new_value&mask) !=(old_value&mask) new_value: current value of the message old_value: last value of the message (initialized with the initial value of the message, updated with new_value if the new message value is not filtered out) **Tags:**atp.EnumerationLiteralIndex=1
maskedNewDiffers X	Pass messages whose masked value is not equal to a specific value x (new_value&mask) != x new_value: current value of the message **Tags:**atp.EnumerationLiteralIndex=2
maskedNewEquals X	Pass messages whose masked value is equal to a specific value x (new_value&mask) == x new_value: current value of the message **Tags:**atp.EnumerationLiteralIndex=3
never	The filter removes all messages. **Tags:**atp.EnumerationLiteralIndex=4
newIsOutside	Pass a message if its value is outside a predefined boundary. (min > new_value) OR (new_value > max) **Tags:**atp.EnumerationLiteralIndex=5

▽

△

Enumeration	**DataFilterTypeEnum**
newIsWithin	Pass a message if its value is within a predefined boundary. min <= new_value <= max **Tags:**atp.EnumerationLiteralIndex=6
oneEveryN	Pass a message once every N message occurrences. Algorithm: occurrence % period == offset Start: occurrence = 0. Each time the message is received or transmitted, occurrence is incremented by 1 after filtering. Length of occurrence is 8 bit (minimum). **Tags:**atp.EnumerationLiteralIndex=7

Table D.42: DataFilterTypeEnum

Class	***DataPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Base class for prototypical roles of any data type.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*ApplicationCompositeElementDataPrototype*, *AutosarDataPrototype*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
swDataDef Props	SwDataDefProps	0..1	aggr	This property allows to specify data definition properties which apply on data prototype level.

Table D.43: DataPrototype

Class	***DiagnosticAbstractDataIdentifierInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class serves as the abstract base class of PortInterfaces dedicated to the access of diagnostic data identifiers on the AUTOSAR adaptive platform. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	DiagnosticDataElementInterface, DiagnosticDataIdentifierGenericInterface, DiagnosticDataIdentifier Interface			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table D.44: DiagnosticAbstractDataIdentifierInterface

Class	**DiagnosticContributionSet**
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticContribution
Note	This meta-class represents a root node of a diagnostic extract. It bundles a given set of diagnostic model elements. The granularity of the DiagonsticContributionSet is arbitrary in order to support the aspect of decentralized configuration, i.e. different contributors can come up with an own DiagnosticContribution Set. **Tags:**atp.recommendedPackage=DiagnosticContributionSets

▽

△

Class	DiagnosticContributionSet			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
common Properties	DiagnosticCommonProps	0..1	aggr	This attribute represents a collection of diagnostic properties that are shared among the entire Diagnostic ContributionSet. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=commonProperties
element	DiagnosticCommonElement	*	ref	This represents a DiagnosticCommonElement considered in the context of the DiagnosticContributionSet **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=element.diagnosticCommonElement, element.variationPoint.shortLabel vh.latestBindingTime=postBuild
serviceTable	DiagnosticServiceTable	*	ref	This represents the collection of DiagnosticServiceTables to be considered in the scope of this Diagnostic ContributionSet. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serviceTable.diagnosticServiceTable, service Table.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table D.45: DiagnosticContributionSet

Class	DiagnosticIndicator			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticIndicator			
Note	Definition of an indicator. **Tags:**atp.recommendedPackage=DiagnosticIndicators			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
type	DiagnosticIndicatorTypeEnum	0..1	attr	Defines the type of the indicator. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime

Table D.46: DiagnosticIndicator

Class	*DiagnosticMapping* (abstract)
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMapping
Note	Abstract element for different kinds of diagnostic mappings.
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable
Subclasses	DiagnosticEventToDebounceAlgorithmMapping, DiagnosticEventToEnableConditionGroupMapping, DiagnosticEventToOperationCycleMapping, DiagnosticEventToTroubleCodeJ1939Mapping, DiagnosticEventToTroubleCodeUdsMapping, DiagnosticFimAliasEventGroupMapping, DiagnosticFimAliasEventMapping, DiagnosticInhibitSourceEventMapping, DiagnosticJ1939SpnMapping, DiagnosticProvidedDataMapping, DiagnosticServiceDataMapping, *DiagnosticSwMapping*, DiagnosticTroubleCodeUdsToClearConditionGroupMapping, DiagnosticTroubleCodeUdsToTroubleCodeObdMapping

▽

△

Class	***DiagnosticMapping*** (abstract)			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table D.47: DiagnosticMapping

Class	***DiagnosticMemoryDestination*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
Note	This abstract meta-class represents a possible memory destination for a diagnostic event.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticMemoryDestinationMirror, DiagnosticMemoryDestinationPrimary, DiagnosticMemoryDestinationUserDefined			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dtcStatusAvailabilityMask	PositiveInteger	0..1	attr	Mask for the supported DTC status bits by the Dem.
eventDisplacementStrategy	DiagnosticEventDisplacementStrategyEnum	0..1	attr	This attribute defines, whether support for event displacement is enabled or not, and which displacement strategy is followed.
maxNumberOfEventEntries	PositiveInteger	0..1	attr	This attribute fixes the maximum number of event entries in the fault memory.
memoryEntryStorageTrigger	DiagnosticMemoryEntryStorageTriggerEnum	0..1	attr	Describes the trigger to allocate an event memory entry.
typeOfFreezeFrameRecordNumeration	DiagnosticTypeOfFreezeFrameRecordNumerationEnum	0..1	attr	This attribute defines the type of assigning freeze frame record numbers for event-specific freeze frame records.

Table D.48: DiagnosticMemoryDestination

Class	**DiagnosticReadDataByIdentifier**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DataByIdentifier			
Note	This represents an instance of the "Read Data by Identifier" diagnostic service. **Tags:**atp.recommendedPackage=DiagnosticDataByIdentifiers			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticDataByIdentifier*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
readClass	DiagnosticReadDataByIdentifierClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticReadDataBy Identifier in the given context.

Table D.49: DiagnosticReadDataByIdentifier

Class	**DiagnosticRoutineNeeds**
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
Note	Specifies the general needs on the configuration of the Diagnostic Communication Manager (Dcm) which are not related to a particular item (e.g. a PID). The main use case is the mapping of service ports to the Dcm which are not related to a particular item.

▽

△

Class	**DiagnosticRoutineNeeds**			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
diagRoutineType	DiagnosticRoutineTypeEnum	0..1	attr	This denotes the type of diagnostic routine which is implemented by the referenced server port.
ridNumber	PositiveInteger	0..1	attr	This represents a routine identifier for the diagnostic routine. This allows to predefine the RID number if the a function developer has received a particular requirement from the OEM or from a standardization body.

Table D.50: DiagnosticRoutineNeeds

Class	**DiagnosticSecurityLevel**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
Note	This meta-class represents the ability to define a security level considered for diagnostic purposes. **Tags:**atp.recommendedPackage=DiagnosticSecurityLevels			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
accessDataRecordSize	PositiveInteger	0..1	attr	This represents the size of the AccessDataRecord used in GetSeed. Unit:byte.
keySize	PositiveInteger	0..1	attr	This represents the size of the security key. Unit: byte.
numFailedSecurityAccess	PositiveInteger	0..1	attr	This represents the number of failed security accesses after which the delay time is activated.
securityDelayTime	TimeValue	0..1	attr	This represents the delay time after a failed security access. Unit: second.
seedSize	PositiveInteger	0..1	attr	This represents the size of the security seed. Unit: byte.

Table D.51: DiagnosticSecurityLevel

Class	***DiagnosticServiceInstance*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommonService			
Note	This represents a concrete instance of a diagnostic service.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticClearDiagnosticInformation, DiagnosticClearResetEmissionRelatedInfo, DiagnosticComControl, DiagnosticControlDTCSetting, DiagnosticCustomServiceInstance, *DiagnosticDataByIdentifier*, DiagnosticDynamicallyDefineDataIdentifier, DiagnosticEcuReset, DiagnosticIOControl, *DiagnosticMemoryByAddress*, DiagnosticReadDTCInformation, DiagnosticReadDataByPeriodicID, DiagnosticRequestControlOfOnBoardDevice, DiagnosticRequestCurrentPowertrainData, DiagnosticRequestEmissionRelatedDTC, DiagnosticRequestEmissionRelatedDTCPermanentStatus, DiagnosticRequestFileTransfer, DiagnosticRequestOnBoardMonitoringTestResults, DiagnosticRequestPowertrainFreezeFrameData, DiagnosticRequestVehicleInfo, DiagnosticResponseOnEvent, DiagnosticRoutineControl, DiagnosticSecurityAccess, DiagnosticSessionControl			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
accessPermission	DiagnosticAccessPermission	0..1	ref	This represents the collection of DiagnosticAccessPermissions that allow for the execution of the referencing DiagnosticServiceInstance..

▽

△

Class	***DiagnosticServiceInstance*** (abstract)			
serviceClass	DiagnosticServiceClass	0..1	ref	This represents the corresponding "class", i.e. this meta-class provides properties that are shared among all instances of applicable sub-classes of DiagnosticService Instance. The subclasses that affected by this pattern implement references to the applicable "class"-role that substantiate this abstract reference. **Stereotypes:** atpAbstract

Table D.52: DiagnosticServiceInstance

Class	**DiagnosticServiceSwMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::ServiceMapping			
Note	This represents the ability to define a mapping of a diagnostic service to a software-component or a basic-software module. If the former is used then this kind of service mapping is applicable for the usage of ClientServerInterfaces. **Tags:**atp.recommendedPackage=DiagnosticServiceMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnosticData Element	DiagnosticDataElement	0..1	ref	This represents a DiagnosticDataElement required to execute the respective diagnostic service in the context of the diagnostic service mapping,
mappedBsw Service Dependency	BswService DependencyIdent	0..1	ref	This is supposed to represent a reference to a Bsw ServiceDependency. the latter is not derived from Referrable and therefore this detour needs to be implemented to still let BswServiceDependency become the target of a reference.
mappedFlatSwc Service Dependency	SwcService Dependency	0..1	ref	This represents the ability to refer to an AtomicSw ComponentType that is available without the definition of how it will be embedded into the component hierarchy.
mappedSwc Service DependencyIn Executable	SwcService Dependency	0..1	iref	This represents the ability to point into the component hiearchy of an adaptive AUTOSAR model (under possible consideration of the rootSoftwareComposition) **Tags:**atp.Status=draft **InstanceRef implemented by:**SwcServiceDependency InExecutableInstanceRef
mappedSwc Service DependencyIn System	SwcService Dependency	0..1	iref	This represents the ability to point into the component hiearchy (under possible consideration of the root SoftwareComposition) **InstanceRef implemented by:**SwcServiceDependency InSystemInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
serviceInstance	DiagnosticService Instance	0..1	ref	This represents the service instance that needs to be considered in this diagnostics service mapping.

Table D.53: DiagnosticServiceSwMapping

Class	DiagnosticTroubleCodeUds			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
Note	This element is used to describe non OBD-relevant DTCs. **Tags:**atp.recommendedPackage=DiagnosticTroubleCodes			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticTroubleCode*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
considerPto Status	Boolean	0..1	attr	This attribute describes the affection of the event by the Dem PTO handling. True: the event is affected by the Dem PTO handling. False: the event is not affected by the Dem PTO handling.
dtcProps	DiagnosticTroubleCode Props	0..1	ref	Defined properties associated with the DemDTC.
eventObd Readiness Group	NameToken	0..1	attr	This attribute specifies the Event OBD Readiness group for PID $01 and PID $41 computation. This attribute is only applicable for emission-related ECUs.
functionalUnit	PositiveInteger	0..1	attr	This attribute specifies a 1-byte value which identifies the corresponding basic vehicle / system function which reports the DTC. This parameter is necessary for the report of severity information.
severity	DiagnosticUdsSeverity Enum	0..1	attr	DTC severity according to ISO 14229-1.
udsDtcValue	PositiveInteger	0..1	attr	Unique Diagnostic Trouble Code value for UDS. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
wwhObdDtc Class	DiagnosticWwhObdDtc ClassEnum	0..1	attr	This attribute is used to identify (if applicable) the corresponding severity class of an WWH-OBD DTC. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime

Table D.54: DiagnosticTroubleCodeUds

Class	DiagnosticUploadDownloadNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the ability to specify needs regarding upload and download by means of diagnostic services.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table D.55: DiagnosticUploadDownloadNeeds

Class	DiagnosticWriteDataByIdentifier
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DataByIdentifier
Note	This represents an instance of the "Write Data by Identifier" diagnostic service. **Tags:**atp.recommendedPackage=DiagnosticDataByIdentifiers
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticDataByIdentifier*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	**DiagnosticWriteDataByIdentifier**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
writeClass	DiagnosticWriteDataBy IdentifierClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticWriteDataBy Identifier in the given context.

Table D.56: DiagnosticWriteDataByIdentifier

Class	**EcuInstance**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	ECUInstances are used to define the ECUs used in the topology. The type of the ECU is defined by a reference to an ECU specified with the ECU resource description. **Tags:**atp.recommendedPackage=EcuInstances			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
associatedCom IPduGroup	ISignalIPduGroup	*	ref	With this reference it is possible to identify which ISignal IPduGroups are applicable for which Communication Connector/ ECU. Only top level ISignalIPduGroups shall be referenced by an EcuInstance. If an ISignalIPduGroup contains other ISignalIPduGroups than these contained ISignalIPdu Groups shall not be referenced by the EcuInstance. Contained ISignalIPduGroups are associated to an Ecu Instance via the top level ISignalIPduGroup.
associated Consumed Provided ServiceInstance Group	ConsumedProvided ServiceInstanceGroup	*	ref	With this reference it is possible to identify which ConsumedProvidedServiceInstanceGroups are applicable for which ECUInstance. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
associatedPdur IPduGroup	PdurIPduGroup	*	ref	With this reference it is possible to identify which PduR IPdu Groups are applicable for which Communication Connector/ ECU.
clientIdRange	ClientIdRange	0..1	aggr	Restriction of the Client Identifier for this Ecu to an allowed range of numerical values. The Client Identifier of the transaction handle is generated by the client RTE for inter-Ecu Client/Server communication.
com Configuration GwTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_Main FunctionRouteSignals of the AUTOSAR COM module in seconds.
com ConfigurationRx TimeBase	TimeValue	0..1	attr	The period between successive calls to Com_Main FunctionRx of the AUTOSAR COM module in seconds.
com ConfigurationTx TimeBase	TimeValue	0..1	attr	The period between successive calls to Com_Main FunctionTx of the AUTOSAR COM module in seconds.
comEnable MDTForCyclic Transmission	Boolean	0..1	attr	Enables for the Com module of this EcuInstance the minimum delay time monitoring for cyclic and repeated transmissions (TransmissionModeTiming has cyclic Timing assigned or eventControlledTiming with numberOf Repetitions > 0).

▽

△

Class	**EcuInstance**			
commController	Communication Controller	1..*	aggr	CommunicationControllers of the ECU. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
connector	Communication Connector	*	aggr	All channels controlled by a single controller. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
doIpConfig	DoIpConfig	0..1	aggr	DoIp configuration on this EcuInstance. **Tags:**atp.Status=draft
ethSwitchPort Group Derivation	Boolean	0..1	attr	Defines whether the derivation of SwitchPortGroups based on VLAN and/or CouplingPort.pncMapping shall be performed for this EcuInstance. If not defined the derivation shall not be done.
pncPrepare SleepTimer	TimeValue	0..1	attr	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.
pnc Synchronous Wakeup	Boolean	0..1	attr	If this parameter is available and set to true then all available PNCs will be woken up as soon as a channel wakeup occurs. This is ensured by adding all PNCs to all channel wakeup sources during upstream mapping.
pnResetTime	TimeValue	0..1	attr	Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests in the EIRA and in the ERA.
sleepMode Supported	Boolean	1	attr	Specifies whether the ECU instance may be put to a "low power mode" • true: sleep mode is supported • false: sleep mode is not supported Note: This flag may only be set to "true" if the feature is supported by both hardware and basic software.
v2xSupported	V2xSupportEnum	0..1	attr	This attribute is used to control the existence of the V2X stack on the given EcuInstance.
wakeUpOver BusSupported	Boolean	1	attr	Driver support for wakeup over Bus.

Table D.57: EcuInstance

Class	**EndToEndTransformationDescription**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	EndToEndTransformationDescription holds these attributes which are profile specific and have the same value for all E2E transformers.			
Base	*ARObject*, *Describable*, *TransformationDescription*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clearFromValid ToInvalid	Boolean	0..1	attr	Clear monitoring window on transition from state Valid to state Invalid.
counterOffset	PositiveInteger	0..1	attr	Offset of the counter in the Data[] array in bits.
crcOffset	PositiveInteger	0..1	attr	Offset of the CRC in the Data[] array in bits.
dataIdMode	DataIdModeEnum	0..1	attr	This attribute describes the inclusion mode that is used to include the implicit two-byte Data ID in the one-byte CRC.
dataIdNibble Offset	PositiveInteger	0..1	attr	Offset of the Data ID nibble in the Data[] array in bits.

▽

△

Class	**EndToEndTransformationDescription**			
e2eProfile Compatibility Props	E2EProfileCompatibility Props	0..1	ref	Reference to additional settings for the E2E state machine.
maxDelta Counter	PositiveInteger	0..1	attr	Maximum allowed difference between two counter values of two consecutively received valid messages. For example, if the receiver gets data with counter 1 and Max DeltaCounter is 3, then at the next reception the receiver can accept Counters with values 2, 3 or 4.
maxErrorState Init	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_INIT.
maxErrorState Invalid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_INVALID.
maxErrorState Valid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_VALID.
maxNoNewOr RepeatedData	PositiveInteger	0..1	attr	The maximum allowed amount of consecutive failed counter checks.
minOkStateInit	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INIT.
minOkState Invalid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INVALID.
minOkState Valid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_VALID.
offset	PositiveInteger	0..1	attr	Offset of the E2E header in the Data[] array in bits.
profileBehavior	EndToEndProfile BehaviorEnum	0..1	attr	Behavior of the check functionality
profileName	NameToken	1	attr	Definition of the E2E profile.
syncCounterInit	PositiveInteger	0..1	attr	Number of checks required for validating the consistency of the counter that shall be received with a valid counter (i.e. counter within the allowed lock-in range) after the detection of an unexpected behavior of a received counter.
upperHeader BitsToShift	PositiveInteger	0..1	attr	This attribute describes the number of upper-header bits to be shifted. value = 0 or not present: shift of upper header is NOT performed. value > 0: the E2E Transformer on the protect-side, takes the first upperHeaderBitsToShift bits from the upper buffer (e.g. SOME/IP header part generated by SOME/IP transformer) and shifts them towards the lower bytes and bits within the Data[] for the length of the E2E header (e.g. 12 bytes in case of E2E Profile 4). This means the shift distance is fixed - it depends on the E2E header size - what is configured here is the number of bits that are to be shifted. This option is defined because the Some/IP header generated by SOME/IP transformer shall be, due to compatibility between non-protected and E2E-protected communication, at the same position, which is before E2E header.
windowSizeInit	PositiveInteger	0..1	attr	Size of the monitoring window of state Init for the E2E state machine.

▽

△

Class	EndToEndTransformationDescription			
windowSize Invalid	PositiveInteger	0..1	attr	Size of the monitoring window of state Invalid for the E2E state machine.
windowSize Valid	PositiveInteger	0..1	attr	Size of the monitoring window of state Valid for the E2E state machine.

Table D.58: EndToEndTransformationDescription

Class	EthernetPhysicalChannel			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	The EthernetPhysicalChannel represents a VLAN or an untagged channel. An untagged channel is modeled as an EthernetPhysicalChannel without an aggregated VLAN.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhysicalChannel*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
network Endpoint	NetworkEndpoint	*	aggr	Collection of NetworkEndpoints that are used in the VLan. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=networkEndpoint.shortName
vlan	VlanConfig	0..1	aggr	VLAN Configuration.

Table D.59: EthernetPhysicalChannel

Class	EventControlledTiming			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication::Timing			
Note	Specification of a event driven sending behavior. The PDU is sent n (numberOfRepeat + 1) times separated by the repetitionPeriod. If numberOfRepeats = 0, then the Pdu is sent just once.			
Base	*ARObject*, *Describable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
numberOf Repetitions	Integer	1	attr	Defines the number of repetitions for the Direct/N-Times transmission mode and the event driven part of Mixed transmission mode.
repetitionPeriod	TimeRangeType	0..1	aggr	The repetitionPeriod specifies the time in seconds that elapses before the pdu can be sent the next time (Minimum repeat gap between two pdus). The repetition Period is optional in case that no repetitions are configured.

Table D.60: EventControlledTiming

Class	***FibexElement*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore			
Note	ASAM FIBEX elements specifying Communication and Topology.			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*BusMirrorChannelMapping*, *CommunicationCluster*, ConsumedProvidedServiceInstanceGroup, CouplingElement, DltMessageCollectionSet, EcuInstance, EthernetWakeupSleepOnDatalineConfigSet, *Frame*, Gateway, GlobalTimeDomain, ISignal, ISignalGroup, ISignalIPduGroup, MachineDesign, Nm Config, *Pdu*, PdurIPduGroup, SecureCommunicationPropsSet, ServiceInstanceCollectionSet, SoAd RoutingGroup, SocketConnectionIpduIdentifierSet, *TpConfig*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	***FibexElement*** (abstract)			
–	–	–	–	–

Table D.61: FibexElement

Class	***Frame*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Data frame which is sent over a communication medium. This element describes the pure Layout of a frame sent on a channel.			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	*AbstractEthernetFrame*, CanFrame, FlexrayFrame, *LinFrame*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
frameLength	Integer	0..1	attr	The used length (in bytes) of the referencing frame. Should not be confused with a static byte length reserved for each frame by some platforms (e.g. FlexRay). The frameLength of zero bytes is allowed. Please consider also TPS_SYST_02255.
pduToFrame Mapping	PduToFrameMapping	*	aggr	A frames layout as a sequence of Pdus. atpVariation: The content of a frame can be variable. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild

Table D.62: Frame

Class	**GlobalTimeDomain**			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime			
Note	This represents the ability to define a global time domain. **Tags:**atp.recommendedPackage=GlobalTimeDomains			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
debounceTime	TimeValue	0..1	attr	Defines the minimum amount of time between two time sync messages are transmitted.
domainId	PositiveInteger	1	attr	This represents the ID of the GlobalTimeDomain used in the network messages sent on behalf of global time management.
gateway	GlobalTimeGateway	*	aggr	A GlobalTimeGateway may exist in the context of a GlobalTimeDomain to actively update the global time information as it is routed from one GlobalTimeDomain to another. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
globalTime CorrectionProps	GlobalTimeCorrection Props	0..1	aggr	Defintion of attributes for rate and offset correction.
globalTime Domain Property	AbstractGlobalTime DomainProps	0..1	aggr	Additional properties of the GlobalTimeDomain. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild

▽

△

Class	**GlobalTimeDomain**			
globalTime Master	GlobalTimeMaster	0..1	aggr	This represents the single master of a GlobalTime Domain. A GlobalTimeDomain may have no GlobalTime Domain.master, e.g. when it gets its time from a GPS receiver. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
globalTimeSub Domain	GlobalTimeDomain	*	ref	By this means it is possible to create a hierarchy of sub Domains where one global time domain can declare one or more other global time domains as its subDomains. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
network SegmentId	NetworkSegment Identification	0..1	aggr	Defines the numerical identification of a GlobalTime sub domain.
offsetTime Domain	GlobalTimeDomain	0..1	ref	Reference to a synchronized time domain this offset time domain is based on. The reference source is the offset time domain. The reference target is the synchronized time domain.
pduTriggering	PduTriggering	0..1	ref	This PduTriggering will be taken to transmit the global time information from a GlobalTimeMaster to a the associated GlobalTimeSlaves. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
slave	GlobalTimeSlave	*	aggr	This represents the collections of slaves of the Global TimeDomain. A GlobalTimeDomain may have no Global TimeDomain.slaves, e.g. when it propagates its time directly to sub domains. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
syncLoss Timeout	TimeValue	0..1	attr	This attribute describes the timeout for the situation that the time synchronization gets lost in the scope of the time domain.

Table D.63: GlobalTimeDomain

Class	***GlobalTimeMaster*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime			
Note	This represents the generic concept of a global time master.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	GlobalTimeCanMaster, GlobalTimeEthMaster, GlobalTimeFrMaster, UserDefinedGlobalTimeMaster			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Connector	Communication Connector	1	ref	The GlobalTimeMaster is bound to the Communication Connector.
immediate ResumeTime	TimeValue	0..1	attr	Defines the minimum time between an "immediate" message and the next periodic message.
isSystemWide GlobalTime Master	Boolean	1	attr	If set to TRUE, the GlobalTimeMaster is supposed to act as the root of global time information.
syncPeriod	TimeValue	1	attr	This represents the period. Unit: seconds

Table D.64: GlobalTimeMaster

Class	***GlobalTimeSlave*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime			
Note	This represents the generic concept of a global time slave.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	GlobalTimeCanSlave, GlobalTimeEthSlave, GlobalTimeFrSlave, UserDefinedGlobalTimeSlave			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Connector	Communication Connector	1	ref	The GlobalTimeSlave is bound to the Communication Connector.
followUp TimeoutValue	TimeValue	0..1	attr	Rx timeout for the follow-up message.
timeLeapFuture Threshold	TimeValue	0..1	attr	Defines the maximum allowed positive difference between the current Local Time Base value and a newly received Global Time Base value.
timeLeap HealingCounter	PositiveInteger	0..1	attr	Defines the required number of updates to the Time Base where the time difference to the previous received value has to remain within the bounds of timeLeapFuture Threshold and timeLeapPastThreshold until that Time Base is considered healed.
timeLeapPast Threshold	TimeValue	0..1	attr	Defines the maximum allowed negative difference between the current Local Time Base value and a newly received Global Time Base value.

Table D.65: GlobalTimeSlave

Class	***Grant*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class serves as the abstract base class for defining specific Grants **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	ComFindServiceGrant, *ComGrant*, ComOfferServiceGrant, *RawDataStreamGrant*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table D.66: Grant

Enumeration	**HandleInvalidEnum**
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication
Note	Strategies of handling the reception of invalidValue.
Literal	***Description***
dontInvalidate	Invalidation is switched off. **Tags:**atp.EnumerationLiteralIndex=0
external Replacement	Replace a received invalidValue. The replacement value is sourced from the externalReplacement. **Tags:**atp.EnumerationLiteralIndex=1
keep	The application software is supposed to handle signal invalidation on RTE API level either by Data ReceiveErrorEvent or check of error code on read access. **Tags:**atp.EnumerationLiteralIndex=2

△

Enumeration	**HandleInvalidEnum**
replace	Replace a received invalidValue. The replacement value is specified by the initValue. **Tags:**atp.EnumerationLiteralIndex=3

Table D.67: HandleInvalidEnum

Class	***IPdu*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	The IPdu (Interaction Layer Protocol Data Unit) element is used to sum up all Pdus that are routed by the PduR.			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Pdu*, *Referrable*			
Subclasses	ContainerIPdu, DcmIPdu, GeneralPurposeIPdu, ISignalIPdu, J1939DcmIPdu, MultiplexedIPdu, NPdu, SecuredIPdu, UserDefinedIPdu			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
containedIPdu Props	ContainedIPduProps	0..1	aggr	Defines whether this IPdu may be collected inside a ContainerIPdu.

Table D.68: IPdu

Class	**IPduTiming**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	AUTOSAR COM provides the possibility to define two different TRANSMISSION MODES for each IPdu. The Transmission Mode of an IPdu that is valid at a specific point in time is selected using the values of the signals that are mapped to this IPdu. For each IPdu a Transmission Mode Selector is defined. The Transmission Mode Selector is calculated by evaluating the conditions for a subset of signals (class TransmissionModeCondition in the System Template). The Transmission Mode Selector is defined to be true, if at least one Condition evaluates to true and is defined to be false, if all Conditions evaluate to false.			
Base	*ARObject*, *Describable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
minimumDelay	TimeValue	0..1	attr	Minimum Delay in seconds between successive transmissions of this I-PDU, independent of the Transmission Mode.
transmission Mode Declaration	TransmissionMode Declaration	0..1	aggr	AUTOSAR COM allows configuring statically two different transmission modes for each I-PDU (True and False). The Transmission Mode Selector evaluates the conditions for a subset of signals and decides the transmission mode. It is possible to switch between the transmission modes during runtime.

Table D.69: IPduTiming

Class	**ISignal**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication
Note	Signal of the Interaction Layer. The RTE supports a "signal fan-out" where the same System Signal is sent in different SignalIPdus to multiple receivers. To support the RTE "signal fan-out" each SignalIPdu contains ISignals. If the same System Signal is to be mapped into several SignalIPdus there is one ISignal needed for each ISignalToIPduMapping. ▽

▽

△

Class	ISignal			
	△ ISignals describe the Interface between the Precompile configured RTE and the potentially Postbuild configured Com Stack (see ECUC Parameter Mapping). In case of the SystemSignalGroup an ISignal shall be created for each SystemSignal contained in the SystemSignalGroup. **Tags:**atp.recommendedPackage=ISignals			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
data Transformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignal. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataTransformation.dataTransformation, dataTransformation.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime
dataTypePolicy	DataTypePolicyEnum	1	attr	With the aggregation of SwDataDefProps an ISignal specifies how it is represented on the network. This representation follows a particular policy. Note that this causes some redundancy which is intended and can be used to support flexible development methodology as well as subsequent integrity checks. If the policy "networkRepresentationFromComSpec" is chosen the network representation from the ComSpec that is aggregated by the PortPrototype shall be used. If the "override" policy is chosen the requirements specified in the PortInterface and in the ComSpec are not fulfilled by the networkRepresentationProps. In case the System Description doesn't use a complete Software Component Description (VFB View) the "legacy" policy can be chosen.
initValue	ValueSpecification	0..1	aggr	Optional definition of a ISignal's initValue in case the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy system signals. This value can be used to configure the Signal's "Init Value". If a full DataMapping exist for the SystemSignal this information may be available from a configured Sender ComSpec and ReceiverComSpec. In this case the initvalues in SenderComSpec and/or ReceiverComSpec override this optional value specification. Further restrictions apply from the RTE specification.
iSignalProps	ISignalProps	0..1	aggr	Additional optional ISignal properties that may be stored in different files. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=iSignalProps
iSignalType	ISignalTypeEnum	0..1	attr	This attribute defines whether this iSignal is an array that results in a UINT8_N / UINT8_DYN ComSignalType in the COM configuration or a primitive type.

▽

△

Class	**ISignal**			
length	Integer	1	attr	Size of the signal in bits. The size needs to be derived from the mapped VariableDataPrototype according to the mapping of primitive DataTypes to BaseTypes as used in the RTE. Indicates maximum size for dynamic length signals. The ISignal length of zero bits is allowed.
network Representation Props	SwDataDefProps	0..1	aggr	Specification of the actual network representation. The usage of SwDataDefProps for this purpose is restricted to the attributes compuMethod and baseType. The optional baseType attributes "memAllignment" and "byteOrder" shall not be used. The attribute "dataTypePolicy" in the SystemTemplate element defines whether this network representation shall be ignored and the information shall be taken over from the network representation of the ComSpec. If "override" is chosen by the system integrator the network representation can violate against the requirements defined in the PortInterface and in the network representation of the ComSpec. In case that the System Description doesn't use a complete Software Component Description (VFB View) this element is used to configure "ComSignalDataInvalid Value" and the Data Semantics.
systemSignal	SystemSignal	1	ref	Reference to the System Signal that is supposed to be transmitted in the ISignal.
timeout Substitution Value	ValueSpecification	0..1	aggr	Defines and enables the ComTimeoutSubstituition for this ISignal.
transformation ISignalProps	TransformationISignal Props	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignal specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignals are described in the TransformationTechnology class.

Table D.70: ISignal

Class	**ISignalGroup**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	SignalGroup of the Interaction Layer. The RTE supports a "signal fan-out" where the same System Signal Group is sent in different SignalIPdus to multiple receivers. An ISignalGroup refers to a set of ISignals that shall always be kept together. A ISignalGroup represents a COM Signal Group. Therefore it is recommended to put the ISignalGroup in the same Package as ISignals (see atp.recommendedPackage) **Tags:**atp.recommendedPackage=ISignalGroup			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

▽

△

Class	**ISignalGroup**			
comBased SignalGroup Transformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignalGroup based on the COMBasedTransformer approach. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=comBasedSignalGroupTransformation.data Transformation, comBasedSignalGroup Transformation.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime
iSignal	ISignal	*	ref	Reference to a set of ISignals that shall always be kept together.
systemSignal Group	SystemSignalGroup	1	ref	Reference to the SystemSignalGroup that is defined on VFB level and that is supposed to be transmitted in the ISignalGroup.
transformation ISignalProps	TransformationISignal Props	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignalGroup specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignal Groups are described in the TransformationTechnology class.

Table D.71: ISignalGroup

Class	**ISignalIPdu**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Represents the IPdus handled by Com. The ISignalIPdu assembled and disassembled in AUTOSAR COM consists of one or more signals. In case no multiplexing is performed this IPdu is routed to/from the Interface Layer. A maximum of one dynamic length signal per IPdu is allowed. **Tags:**atp.recommendedPackage=Pdus			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *IPdu*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Pdu*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iPduTiming Specification	IPduTiming	0..1	aggr	Timing specification for Com IPdus (Transmission Modes). This information is mandatory for the sender in a System Extract. This information may be omitted on receivers in a System Extract. atpVariation: The timing of a Pdu can vary. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
iSignalToPdu Mapping	ISignalToIPduMapping	*	aggr	Definition of SignalToIPduMappings included in the Signal IPdu. atpVariation: The content of a PDU can be variable. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
pduCounter	SignalIPduCounter	0..1	aggr	An included Pdu counter is used to ensure that a sequence of Pdus is maintained. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime

▽

△

Class	**ISignalIPdu**			
pduReplication	SignalIPduReplication	0..1	aggr	Pdu Replication is a form of redundancy where the data content of one ISignalIPdu (source) is transmitted inside a set of replica ISignalIPdus. These ISignalIPdus (copies) have different Pdu IDs, identical PduCounters, identical data content and are transmitted with the same frequency. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
unusedBit Pattern	Integer	1	attr	AUTOSAR COM and AUTOSAR IPDUM are filling not used areas of an IPDU with this bit-pattern. This attribute is mandatory to avoid undefined behavior. This byte-pattern will be repeated throughout the IPdu.

Table D.72: ISignalIPdu

Class	**ISignalPort**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Connectors reception or send port on the referenced channel referenced by an ISignalTriggering. If different timeouts or DataFilters for ISignals need to be specified several ISignalPorts may be created.			
Base	*ARObject*, *CommConnectorPort*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataFilter	DataFilter	0..1	aggr	Optional specification of a signal COM filter at the receiver side in case that the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy system signals. If a full DataMapping exist for the SystemSignal this information may be available from a configured ReceiverComSpec. In this case the ReceiverComSpec overrides this optional specification.
firstTimeout	TimeValue	0..1	attr	• ISignalPort with communicationDirection = in: Optional first timeout value in seconds for the reception of the ISignal. • ISignalPort with communicationDirection = out: Optional first timeout value in seconds for transmission deadline monitoring.
handleInvalid	HandleInvalidEnum	0..1	attr	This attribute defines how invalidation is applied to the ISignals received in the context of this ISignalPort.
timeout	TimeValue	0..1	attr	• ISignalPort with communicationDirection = in: Optional timeout value in seconds for the reception of the ISignal in case the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy system signals. If a full DataMapping exist for the SystemSignal this information may be available from a configured Receiver ComSpec, in this case the timeout value in ReceiverCom Spec overrides this optional timeout specification. • ISignalPort with communicationDirection = out: Optional timeout value in seconds for transmission deadline monitoring in case the System Description doesn't use a complete Software Component Description ▽

▽

△

Class	ISignalPort			
				△ (VFB View). This supports the inclusion of legacy system signals. If a full DataMapping exist for the SystemSignal this information may be available from a configured SenderComSpec, in this case the timeout value in Sender ComSpec overrides this optional timeout specification.

Table D.73: ISignalPort

Class	ISignalToIPduMapping			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	An ISignalToIPduMapping describes the mapping of ISignals to ISignalIPdus and defines the position of the ISignal within an ISignalIPdu.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
iSignal	ISignal	0..1	ref	Reference to a ISignal that is mapped into the ISignal IPdu. Each ISignal contained in the ISignalGroup shall be mapped into an IPdu by an own ISignalToIPduMapping. The references to the ISignal and to the ISignalGroup in an ISignalToIPduMapping are mutually exclusive.
iSignalGroup	ISignalGroup	0..1	ref	Reference to an ISignalGroup that is mapped into the SignalIPdu. If an ISignalToIPduMapping for an ISignal Group is defined, only the UpdateIndicationBitPosition and the transferProperty is relevant. The startPosition and the packingByteOrder shall be ignored. Each ISignal contained in the ISignalGroup shall be mapped into an IPdu by an own ISignalToIPduMapping. The references to the ISignal and to the ISignalGroup in an ISignalToIPduMapping are mutually exclusive.
packingByte Order	ByteOrderEnum	0..1	attr	This parameter defines the order of the bytes of the signal and the packing into the SignalIPdu. The byte ordering "Little Endian" (MostSignificantByteLast), "Big Endian" (MostSignificantByteFirst) and "Opaque" can be selected. For opaque data endianness conversion shall be configured to Opaque. The value of this attribute impacts the absolute position of the signal into the SignalIPdu (see the startPosition attribute description). For an ISignalGroup the packingByteOrder is irrelevant and shall be ignored.
startPosition	Integer	0..1	attr	This parameter is necessary to describe the bitposition of a signal within an SignalIPdu. It denotes the least significant bit for "Little Endian" and the most significant bit for "Big Endian" packed signals within the IPdu (see the description of the packingByteOrder attribute). In AUTOSAR the bit counting is always set to "sawtooth" and the bit order is set to "Decreasing". The bit counting in byte 0 starts with bit 0 (least significant bit). The most significant bit in byte 0 is bit 7. Please note that the way the bytes will be actually sent on the bus does not impact this representation: they will always be seen by the software as a byte array. If a mapping for the ISignalGroup is defined, this attribute is irrelevant and shall be ignored.

▽

△

Class	**ISignalToIPduMapping**			
transferProperty	TransferPropertyEnum	0..1	attr	Defines how the referenced ISignal contributes to the send triggering of the ISignalIPdu.
update IndicationBit Position	Integer	0..1	attr	The UpdateIndicationBit indicates to the receivers that the signal (or the signal group) was updated by the sender. Length is always one bit. The UpdateIndicationBitPosition attribute describes the position of the update bit within the SignalIPdu. For Signals of a ISignalGroup this attribute is irrelevant and shall be ignored. Note that the exact bit position of the updateIndicationBit Position is linked to the value of the attribute packingByte Order because the method of finding the bit position is different for the values mostSignificantByteFirst and most SignificantByteLast. This means that if the value of packingByteOrder is changed while the value of update IndicationBitPosition remains unchanged the exact bit position of updateIndicationBitPosition within the enclosing ISignalIPdu still undergoes a change. This attribute denotes the least significant bit for "Little Endian" and the most significant bit for "Big Endian" packed signals within the IPdu (see the description of the packingByteOrder attribute). In AUTOSAR the bit counting is always set to "sawtooth" and the bit order is set to "Decreasing". The bit counting in byte 0 starts with bit 0 (least significant bit). The most significant bit in byte 0 is bit 7.

Table D.74: ISignalToIPduMapping

Class	**ISignalTriggering**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	A ISignalTriggering allows an assignment of ISignals to physical channels.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iSignal	ISignal	0..1	ref	This reference shall be used if an ISignal is transported on the PhysicalChannel. This reference forms an XOR relationship with the ISignalTriggering-ISignalGroup reference.
iSignalGroup	ISignalGroup	0..1	ref	This reference shall be used if an ISignalGroup is transported on the PhysicalChannel. This reference forms an XOR relationship with the ISignal Triggering-ISignal reference.
iSignalPort	ISignalPort	*	ref	References to the ISignalPort on every ECU of the system which sends and/or receives the ISignal. References for both the sender and the receiver side shall be included when the system is completely defined.

Table D.75: ISignalTriggering

Class	**IamModuleInstantiation**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement
Note	This meta-class represents the ability to define a definition of an IAM instantiation. **Tags:**atp.Status=draft

▽

△

Class	**IamModuleInstantiation**			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *NonOsModule Instantiation*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
grant	Grant	*	ref	This reference identifies the applicable Grants for this Iam ModuleInstantiation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=grant atp.Status=draft
localCom AccessControl Enabled	Boolean	0..1	attr	This switch activates the policy enforcement in Communication Management on local applications.
remoteAccess ControlEnabled	Boolean	0..1	attr	This switch activates the check of the remote subject.

Table D.76: IamModuleInstantiation

Class	***Identifiable*** (abstract)
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*
Subclasses	ARPackage, *AbstractDoIpLogicAddressProps*, *AbstractEvent*, *AbstractImplementationDataTypeElement*, *AbstractSecurityEventFilter*, *AbstractSecurityIdsmInstanceFilter*, *AbstractServiceInstance*, *Abstract SignalBasedToISignalTriggeringMapping*, *AdaptiveModuleInstantiation*, AdaptiveSwcInternalBehavior, ApplicationEndpoint, ApplicationError, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, Autosar OperationArgumentInstance, AutosarVariableInstance, *BuildActionEntity*, BuildActionEnvironment, Chapter, CheckpointTransition, ClassContentConditional, ClientIdDefinition, ClientServerOperation, Code, *CollectableElement*, ComManagementMapping, *CommConnectorPort*, *Communication Connector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEventGroup, Coupling Port, *CouplingPortStructuralElement*, CryptoCertificate, CryptoKeySlot, CryptoProvider, *CryptoService Mapping*, DataPrototypeGroup, DataTransformation, DependencyOnArtifact, DeterministicClient ResourceNeeds, *DiagEventDebounceAlgorithm*, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticFunctionInhibitSource, *DiagnosticRoutineSubfunction*, DltArgument, DltLogChannel, Dlt Message, DoIpInterface, DoIpLogicAddress, DoIpRoutingActivation, E2EProfileConfiguration, End2End EventProtectionProps, EndToEndProtection, EthernetWakeupSleepOnDatalineConfig, EventMapping, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FMFeatureMapAssertion, FMFeature MapCondition, FMFeatureMapElement, FMFeatureRelation, FMFeatureRestriction, FMFeatureSelection, FieldMapping, FireAndForgetMapping, *FrameTriggering*, GeneralParameter, GlobalSupervision, Global TimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HealthChannel*, *HeapUsage*, HwAttributeDef, Hw AttributeLiteralDef, HwPin, HwPinGroup, IPSecRule, IPv6ExtHeaderFilterList, ISignalToIPduMapping, I SignalTriggering, *IdentCaption*, InterfaceMapping, InternalTriggeringPoint, Keyword, LifeCycleState, Linker, MacMulticastGroup, McDataInstance, MemorySection, MethodMapping, ModeDeclaration, Mode DeclarationMapping, ModeSwitchPoint, NetworkEndpoint, *NmCluster*, *NmNode*, *PackageableElement*, ParameterAccess, PduToFrameMapping, PduTriggering, *PersistencyDeploymentElement*, *Persistency InterfaceElement*, *PhmSupervision*, *PhysicalChannel*, PortGroup, *PortInterfaceMapping*, PossibleError Reaction, ProcessDesignToMachineDesignMapping, ProcessToMachineMapping, Processor, Processor Core, PskIdentityToKeySlotMapping, RecoveryNotification, ResourceConsumption, ResourceGroup, *RestAbstractEndpoint*, RestElementDef, RestResourceDef, RootSwClusterDesignComponentPrototype, RootSwComponentPrototype, RootSwCompositionPrototype, RptComponent, RptContainer, Rpt ExecutableEntity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptServicePoint, *Sdg Attribute*, SdgClass, SecOcJobMapping, SecOcJobRequirement, *SecureComProps*, Secure CommunicationAuthenticationProps, *SecureCommunicationDeployment*, SecureCommunication FreshnessProps, SecurityEventContextProps, *ServiceEventDeployment*, *ServiceFieldDeployment*, ▽

▽

△

Class	**Identifiable** (abstract)			
	△ ServiceInstanceToSignalMapping, *ServiceInterfaceElementMapping*, ServiceInterfaceElementSecure ComConfig, ServiceInterfaceMapping, *ServiceMethodDeployment*, *ServiceNeeds*, SignalService TranslationEventProps, SignalServiceTranslationProps, SocketAddress, SoftwarePackageStep, Someip EventGroup, SomeipProvidedEventGroup, SomeipTpChannel, *SpecElementReference*, *StackUsage*, StartupConfig, StaticSocketConnection, StructuredReq, SupervisionCheckpoint, SwGenericAxisParam Type, SwServiceArg, SwcServiceDependency, SystemMapping, SystemMemoryUsage, *TimeBase Resource*, TimingCondition, *TimingConstraint*, *TimingDescription*, TimingExtensionResource, Timing ModeInstance, TlsCryptoCipherSuite, TlsJobMapping, Topic1, TpAddress, TraceableTable, Traceable Text, *TracedFailure*, *TransformationProps*, TransformationPropsToServiceInterfaceElementMapping, TransformationTechnology, Trigger, UcmDescription, UcmStep, VariableAccess, VariationPointProxy, VehicleRolloutStep, ViewMap, VlanConfig			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Tags:**xml.sequenceOffset=-40
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. **Tags:**xml.sequenceOffset=-25
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints. **Tags:**xml.sequenceOffset=-50
desc	MultiLanguageOverview Paragraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question. More elaborate documentation, (in particular how the object is built or used) should go to "introduction". **Tags:**xml.sequenceOffset=-60
introduction	DocumentationBlock	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. **Tags:**xml.sequenceOffset=-30
uuid	String	0..1	attr	The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The ▽

▽

△

Class	***Identifiable*** (abstract)			
				△ uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp. **Tags:**xml.attribute=true

Table D.77: Identifiable

Class	**ImplementationDataType**			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code. **Tags:**atp.recommendedPackage=ImplementationDataTypes			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicArray SizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.
isStructWith Optional Element	Boolean	0..1	attr	This attribute is only valid if the attribute category is set to STRUCTURE. If set to True, this attribute indicates that the ImplementationDataType has been created with the intention to define at least one element of the structure as optional.
subElement (ordered)	ImplementationData TypeElement	*	aggr	Specifies an element of an array, struct, or union data type. The aggregation of ImplementionDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a Implementation DataType representing a structure. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the Implementation DataType. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=symbolProps.shortName
typeEmitter	NameToken	0..1	attr	This attribute is used to control which part of the AUTOSAR toolchain is supposed to trigger data type definitions.

Table D.78: ImplementationDataType

Class	**ImplementationDataTypeElement**
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes

▽

△

Class	**ImplementationDataTypeElement**			
Note	Declares a data object which is locally aggregated. Such an element can only be used within the scope where it is aggregated. This element either consists of further subElements or it is further defined via its swDataDefProps. There are several use cases within the system of ImplementationDataTypes fur such a local declaration: • It can represent the elements of an array, defining the element type and array size • It can represent an element of a struct, defining its type • It can be the local declaration of a debug element.			
Base	*ARObject*, *AbstractImplementationDataTypeElement*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arrayImplPolicy	ArrayImplPolicyEnum	0..1	attr	This attribute controls the implementation of the payload of an array. It shall only be used if the enclosing ImplementationDataType constitutes an array.
arraySize	PositiveInteger	0..1	attr	The existence of this attributes (if bigger than 0) defines the size of an array and declares that this Implementation DataTypeElement represents the type of each single array element. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
arraySize Handling	ArraySizeHandling Enum	0..1	attr	The way how the size of the array is handled in case of a variable size array.
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls the meaning of the value of the array size.
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing ImplementationDataTypeElement as optional. This means that, at runtime, the ImplementationDataType Element may or may not have a valid value and shall therefore be ignored. The underlying runtime software provides means to set the CppImplementationDataTypeElement as not valid at the sending end of a communication and determine its validity at the receiving end.
subElement (ordered)	ImplementationData TypeElement	*	aggr	Element of an array, struct, or union in case of a nested declaration (i.e. without using "typedefs"). The aggregation of ImplementionDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a Implementation DataType representing a structure. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
swDataDef Props	SwDataDefProps	0..1	aggr	The properties of this ImplementationDataTypeElement.

Table D.79: ImplementationDataTypeElement

Class	**MacMulticastConfiguration**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology
Note	References a per cluster globally defined MAC-Multicast-Group.
Base	*ARObject*, *NetworkEndpointAddress*

▽

△

Class	**MacMulticastConfiguration**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
macMulticast Group	MacMulticastGroup	1	ref	Reference to a macMulticastGroup.

Table D.80: MacMulticastConfiguration

Class	**MultiplexedIPdu**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	A MultiplexedPdu (i.e. NOT a COM I-PDU) contains a DynamicPart, an optional StaticPart and a selector Field. In case of multiplexing this IPdu is routed between the Pdu Multiplexer and the Interface Layer. A multiplexer is used to define variable parts within an IPdu that may carry different signals. The receivers of such a IPdu can determine which signalPdus are transmitted by evaluating the selector field, which carries a unique selector code for each sub-part. **Tags:**atp.recommendedPackage=Pdus			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *IPdu*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Pdu*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicPart	DynamicPart	0..1	aggr	According to the value of the selector field some parts of the IPdu have a different layout. In a complete System Description a MultiplexedIPdu shall contain a Dynamic Part. The following use cases support the multiplicity to be 0..1: • If a MultiplexedIPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedIPdu doesn't need to be described in the System Extract/Ecu Extract. • If a MultiplexedIPdu is received by an ECU which is only interested in the static part of the MultiplexedIPdu then the dynamicPart does not need to be described in the System Extract/Ecu Extract. atpVariation: Content of a multiplexed PDU can vary. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
selectorField ByteOrder	ByteOrderEnum	0..1	attr	This attribute defines the order of the bytes of the selector Field and the packing into the MultiplexedIPdu. Please consider that [constr_3247] and [constr_3223] are restricting the usage of this attribute. In a complete System Description this attribute is mandatory. If a MultiplexedPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedPdu doesn't need to be described in the System Extract/Ecu Extract. To support this use case the multiplicity is set to 0..1.

▽

△

Class	**MultiplexedIPdu**			
selectorField Length	Integer	0..1	attr	The size in bits of the selector field shall be configurable in a range of 1-16 bits. In a complete System Description this attribute is mandatory. If a MultiplexedPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedPdu doesn't need to be described in the System Extract/Ecu Extract. To support this use case the multiplicity is set to 0..1.
selectorField StartPosition	Integer	0..1	attr	This parameter is necessary to describe the position of the selector field within the IPdu. Note that the absolute position of the selectorField in the MultiplexedIPdu is determined by the definition of the selectorFieldByteOrder attribute of the Multiplexed Pdu. If Big Endian is specified, the start position indicates the bit position of the most significant bit in the IPdu. If Little Endian is specified, the start position indicates the bit position of the least significant bit in the IPdu. In AUTOSAR the bit counting is always set to "sawtooth" and the bit order is set to "Decreasing". The bit counting in byte 0 starts with bit 0 (least significant bit). The most significant bit in byte 0 is bit 7. In a complete System Description this attribute is mandatory. If a MultiplexedPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedPdu doesn't need to be described in the System Extract/Ecu Extract. To support this use case the multiplicity is set to 0..1.
staticPart	StaticPart	0..1	aggr	The static part of the multiplexed IPdu is the same regardless of the selector field. The static part is optional. atpVariation: Content of a multiplexed PDU can vary. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
triggerMode	TriggerMode	0..1	attr	IPduM can be configured to send a transmission request for the new multiplexed IPdu to the PDU-Router because of the trigger conditions/ modes that are described in the TriggerMode enumeration. In a complete System Description this attribute is mandatory. If a MultiplexedPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedPdu doesn't need to be described in the System Extract/Ecu Extract. To support this use case the multiplicity is set to 0..1.
unusedBit Pattern	Integer	0..1	attr	AUTOSAR COM and AUTOSAR IPDUM are filling not used areas of an IPdu with this bit-pattern. This attribute is mandatory to avoid undefined behavior. This byte-pattern will be repeated throughout the IPdu. In a complete System Description this attribute is mandatory. If a MultiplexedPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedPdu doesn't need to be described in the System Extract/Ecu Extract. To support this use case the multiplicity is set to 0..1.

Table D.81: MultiplexedIPdu

Primitive	**NameToken**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This is an identifier as used in xml, e.g. xml-names. Basic difference to Identifier is the fact that it can contain "-". **Tags:** xml.xsd.customType=NMTOKEN-STRING xml.xsd.type=NMTOKEN

Table D.82: NameToken

Class	**NotAvailableValueSpecification**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This meta-class provides the ability to specify a ValueSpecification to state that the respective element is not available. This ability is needed to support the existence of ApplicationRecordElements where attribute isOptional ist set to the value True. **Tags:**atp.Status=draft			
Base	*ARObject*, *ValueSpecification*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defaultPattern	PositiveInteger	0..1	attr	The content of this attribute shall be used to initialize gaps in the memory occupied by a structured data type in the case that an NotAvailableValueSpecification is used. Note that this pattern is only applied during initialization!

Table D.83: NotAvailableValueSpecification

Class	***PPortComSpec*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes of a provided PortPrototype. This class will contain attributes that are valid for all kinds of provide ports, independent of client-server or sender-receiver communication patterns.			
Base	*ARObject*			
Subclasses	ModeSwitchSenderComSpec, NvProvideComSpec, ParameterProvideComSpec, *SenderComSpec*, ServerComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table D.84: PPortComSpec

Class	**PPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port providing a certain port interface.			
Base	*ARObject*, *AbstractProvidedPortPrototype*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
provided Interface	PortInterface	0..1	tref	The interface that this port provides. **Stereotypes:** isOfType

Table D.85: PPortPrototype

Class	**PPortPrototypeInSoftwareClusterDesignInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInSoftwareClusterDesignInstanceRef			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AbstractPortPrototypeInSoftwareClusterDesignInstanceRef*, *AtpInstanceRef*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	SoftwareClusterDesign	0..1	ref	**Stereotypes:** atpDerived **Tags:**atp.Status=draft
contextRootSw ClusterDesign Component Prototype	RootSwClusterDesign ComponentPrototype	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10
contextSw Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetPPort Prototype	PPortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30

Table D.86: PPortPrototypeInSoftwareClusterDesignInstanceRef

Class	**PRPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This kind of PortPrototype can take the role of both a required and a provided PortPrototype.			
Base	*ARObject*, *AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*, *AtpBlueprintable*, *Atp Feature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
provided Required Interface	PortInterface	0..1	tref	This represents the PortInterface used to type the PRPort Prototype **Stereotypes:** isOfType

Table D.87: PRPortPrototype

Class	***PackageableElement*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
Note	This meta-class specifies the ability to be a member of an AUTOSAR package.			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*ARElement*, EnumerationMappingTable, *FibexElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table D.88: PackageableElement

Class	**PassThroughSwConnector**
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition
Note	This kind of SwConnector can be used inside a CompositionSwComponentType to connect two delegation PortPrototypes.

▽

△

Class	PassThroughSwConnector			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SwConnector*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
providedOuterPort	AbstractProvidedPortPrototype	0..1	ref	This represents the provided outer delegation Port Prototype of the PassThroughSwConnector.
requiredOuterPort	AbstractRequiredPortPrototype	0..1	ref	This represents the required outer delegation Port Prototype of the PassThroughSwConnector.

Table D.89: PassThroughSwConnector

Class	**Pdu** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Collection of all Pdus that can be routed through a bus interface.			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	GeneralPurposePdu, *IPdu*, NmPdu, UserDefinedPdu			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
hasDynamicLength	Boolean	0..1	attr	This attribute defines whether the Pdu has dynamic length (true) or not (false). Please note that the usage of this attribute is restricted by [constr_3448].
length	Integer	0..1	attr	Pdu length in bytes. In case of dynamic length IPdus (containing a dynamical length signal), this value indicates the maximum data length. It should be noted that in former AUTOSAR releases (Rel 2.1, Rel 3.0, Rel 3.1, Rel 4.0 Rev. 1) this parameter was defined in bits. The Pdu length of zero bytes is allowed.

Table D.90: Pdu

Class	<<atpPrototype>> **PduToFrameMapping**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	A PduToFrameMapping defines the composition of Pdus in each frame.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
packingByteOrder	ByteOrderEnum	1	attr	This attribute defines the order of the bytes of the Pdu and the packing into the Frame. Please consider that [constr_3246] and [constr_3222] are restricting the usage of this attribute.
pdu	Pdu	1	ref	Reference to a I-Pdu, N-Pdu or NmPdu that is transmitted in the Frame.
startPosition	Integer	1	attr	This attribute describes the bitposition of a Pdu within a Frame. Please note that the absolute position of the Pdu in the Frame is determined by the definition of the packingByte Order attribute. If Big Endian is specified, the start position indicates the bit position of the most significant bit in the Frame. If Little Endian is specified, the start position indicates the bit position of the least significant bit in the ▽

▽

△

Class	<<atpPrototype>> **PduToFrameMapping**			
				△ Frame. The bit counting in byte 0 starts with bit 0 (least significant bit). The most significant bit in byte 0 is bit 7. The Pdus are byte aligned in a Frame and only the values 0, 8, 16, 24,... (for little endian) and 7, 15, 23, ... (for big endian) are allowed.
update IndicationBit Position	Integer	0..1	attr	Indication to the receivers that the corresponding Pdu was updated by the sender. This attribute describes the position of the update bit in the frame that aggregates this PDUToFrameMapping. Length is always one bit. Note that the exact bit position of the updateIndicationBit Position is linked to the value of the attribute packingByte Order because the method of finding the bit position is different for the values mostSignificantByteFirst and most SignificantByteLast. This means that if the value of packingByteOrder is changed while the value of update IndicationBitPosition remains unchanged the exact bit position of updateIndicationBitPosition within the enclosing Frame still undergoes a change. This attribute denotes the least significant bit for "Little Endian" and the most significant bit for "Big Endian" packed signals within the IPdu (see the description of the packingByteOrder attribute). In AUTOSAR the bit counting is always set to "sawtooth" and the bit order is set to "Decreasing". The bit counting in byte 0 starts with bit 0 (least significant bit). The most significant bit in byte 0 is bit 7.

Table D.91: PduToFrameMapping

Class	**PduTriggering**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	The PduTriggering describes on which channel the IPdu is transmitted. The Pdu routing by the PduR is only allowed for subclasses of IPdu. Depending on its relation to entities such channels and clusters it can be unambiguously deduced whether a fan-out is handled by the Pdu router or the Bus Interface. If the fan-out is specified between different clusters it shall be handled by the Pdu Router. If the fan-out is specified between different channels of the same cluster it shall be handled by the Bus Interface.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
iPdu	Pdu	1	ref	Reference to the Pdu for which the PduTriggering is defined. One I-Pdu can be triggered on different channels (PduR fan-out). The Pdu routing by the PduR is only allowed for subclasses of IPdu. Nevertheless is the reference to the Pdu element necessary since the PduTriggering element is also used to specify the sending and receiving connections to Ecu Ports.
iPduPort	IPduPort	*	ref	References to the IPduPort on every ECU of the system which sends and/or receives the I-PDU. References for both the sender and the receiver side shall be included when the system is completely defined.

▽

△

Class	PduTriggering			
iSignal Triggering	ISignalTriggering	*	ref	This reference provides the relationship to the ISignal Triggerings that are implemented by the PduTriggering. The reference is optional since no ISignalTriggering can be defined for DCM and Multiplexed Pdus. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
secOcCrypto Mapping	SecOcCryptoService Mapping	0..1	ref	This reference identifies the crypto profile applicable to the usage (send, receive) of the also referenced Secured IPdu. Obviously, this reference is only applicable if the Pdutriggering also references a SecuredIPdu in the role i Pdu.
triggerIPduSend Condition	TriggerIPduSend Condition	*	aggr	Defines the trigger for the Com_TriggerIPDUSend API call. Only if all defined TriggerIPduSendConditions evaluate to true (AND associated) the Com_Trigger IPDUSend API shall be called.

Table D.92: PduTriggering

Class	*PhmAbstractRecoveryNotificationInterface* (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This abstract meta-class provides the abstract ability to define a PortInterface for the Recovery Notification by Platform Health Management. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PlatformHealthManagementInterface*, *Port Interface*, *Referrable*			
Subclasses	PhmHealthChannelRecoveryNotificationInterface, PhmSupervisionRecoveryNotificationInterface			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table D.93: PhmAbstractRecoveryNotificationInterface

Class	PhysicalDimension			
Package	M2::MSR::AsamHdo::Units			
Note	This class represents a physical dimension. If the physical dimension of two units is identical, then a conversion between them is possible. The conversion between units is related to the definition of the physical dimension. Note that the equivalence of the exponents does not per se define the convertibility. For example Energy and Torque share the same exponents (Nm). Please note further the value of an exponent does not necessarily have to be an integer number. It is also possible that the value yields a rational number, e.g. to compute the square root of a given physical quantity. In this case the exponent value would be a rational number where the numerator value is 1 and the denominator value is 2. **Tags:**atp.recommendedPackage=PhysicalDimensions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

▽

△

Class	**PhysicalDimension**			
currentExp	Numerical	0..1	attr	This attribute represents the exponent of the physical dimension "electric current". **Tags:**xml.sequenceOffset=50
lengthExp	Numerical	0..1	attr	The exponent of the physical dimension "length". **Tags:**xml.sequenceOffset=20
luminous IntensityExp	Numerical	0..1	attr	The exponent of the physical dimension "luminous intensity". **Tags:**xml.sequenceOffset=80
massExp	Numerical	0..1	attr	The exponent of the physical dimension "mass". **Tags:**xml.sequenceOffset=30
molarAmount Exp	Numerical	0..1	attr	The exponent of the physical dimension "quantity of substance". **Tags:**xml.sequenceOffset=70
temperatureExp	Numerical	0..1	attr	The exponent of the physical dimension "temperature". **Tags:**xml.sequenceOffset=60
timeExp	Numerical	0..1	attr	The exponent of the physical dimension "time". **Tags:**xml.sequenceOffset=40

Table D.94: PhysicalDimension

Class	**PlatformModuleEthernetEndpointConfiguration**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class defines the attributes for the configuration of a port, protocol type and IP address of the communication on a VLAN. **Tags:** atp.Status=draft atp.recommendedPackage=PlatformModuleEndpointConfigurations			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PlatformModuleEndpointConfiguration*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Connector	EthernetCommunication Connector	0..1	ref	Reference to the CommunicationConnector (VLAN) for which the network configuration is defined. **Tags:**atp.Status=draft
ipv4MulticastIp Address	Ip4AddressString	0..1	attr	Multicast IPv4 Address to which the message will be transmitted.
ipv6MulticastIp Address	Ip6AddressString	0..1	attr	Multicast IPv6 Address to which the message will be transmitted.
tcpPort	PositiveInteger	0..1	attr	This attribute allows to configure a tcp port number.
udpPort	PositiveInteger	0..1	attr	This attribute allows to configure a udp port number.

Table D.95: PlatformModuleEthernetEndpointConfiguration

Class	**PortGroup**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Group of ports which share a common functionality , e.g. need specific network resources. This information shall be available on the VFB level in order to delegate it properly via compositions. When propagated into the ECU extract, this information is used as input for the configuration of Services like the Communication Manager. A PortGroup is defined locally in a component (which can be a composition) and refers to the "outer" ports belonging to the group as well as to the "inner" groups which propagate this group into the components which are part of a composition. A PortGroup within an atomic SWC cannot be linked to inner groups.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
innerGroup	PortGroup	*	iref	Links a PortGroup in a composition to another PortGroup, that is defined in a component which is part of this CompositionSwComponentType. **InstanceRef implemented by:**InnerPortGroupIn CompositionInstanceRef
outerPort	PortPrototype	*	ref	Outer PortPrototype of this AtomicSwComponentType which belongs to the group. A port can belong to several groups or to no group at all. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime

Table D.96: PortGroup

Class	***PortPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/ server communication.
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port.
ioHwAbstraction Server Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port.
modePort Annotation	ModePortAnnotation	*	aggr	Annotations on this mode port.
nvDataPort Annotation	NvDataPortAnnotation	*	aggr	Annotations on this non voilatile data port.
parameterPort Annotation	ParameterPort Annotation	*	aggr	Annotations on this parameter port.
portPrototype Props	PortPrototypeProps	0..1	aggr	This attribute allows for the definition of further qualification of the semantics of a PortPrototype. **Tags:**atp.Status=draft
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication.

▽

△

Class	***PortPrototype*** (abstract)			
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table D.97: PortPrototype

Class	**ProvidedServiceInstance**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances			
Note	Service instances that are provided by the ECU that is connected via the ApplicationEndpoint to a CommunicationConnector.			
Base	*ARObject*, *AbstractServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
autoAvailable	Boolean	0..1	attr	Defines that this ProvidedServiceInstance shall be offered by the service discovery at ECU start.
eventHandler	EventHandler	*	aggr	Collection of event groups provided by the Provided ServiceInstance **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
instance Identifier	PositiveInteger	0..1	attr	Instance identifier. Can be used for e.g. service discovery to identify the instance of the service.
loadBalancing Priority	PositiveInteger	0..1	attr	Defines the value to be used for load balancing priority in the service offer. Lower value means higher priority.
loadBalancing Weight	PositiveInteger	0..1	attr	Defines the value to be used for load balancing weight in the service offer. Higher value means higher probability to be chosen.
localUnicast Address	ApplicationEndpoint	0..2	ref	The local address over which the PSI is provided (udp, tcp or both). **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
minorVersion	PositiveInteger	0..1	attr	Minor Version of the Service that is provided by this ProvidedServiceInstance.
priority	PositiveInteger	0..1	attr	Defines the frame priority where values from 0 (best effort) to 7 (highest) are allowed.
remoteUnicast Address	ApplicationEndpoint	*	ref	This reference defines the remote addresses of service consumers. This reference shall ONLY be used if the remote address of the clients is determined from the configuration and not at runtime. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
sdServerConfig	SdServerConfig	0..1	aggr	Service Discovery Server configuration. **Tags:**atp.Status=obsolete
sdServerTimer Config	SomeipSdServer ServiceInstanceConfig	0..1	ref	Server specific configuration settings relevant for the SOME/IP service discovery. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
serviceIdentifier	PositiveInteger	0..1	attr	This attribute represents the ability to describe the SOME/IP service ID that is offered.

Table D.98: ProvidedServiceInstance

Class	**RPortComSpec** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes of a required PortPrototype. This class will contain attributes that are valid for all kinds of require-ports, independent of client-server or sender-receiver communication patterns.			
Base	*ARObject*			
Subclasses	ClientComSpec, CryptoRPortComSpec, ModeSwitchReceiverComSpec, NvRequireComSpec, ParameterRequireComSpec, PersistencyDataRequiredComSpec, *ReceiverComSpec*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table D.99: RPortComSpec

Class	**RPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port requiring a certain port interface.			
Base	*ARObject*, *AbstractRequiredPortPrototype*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
requiredInterface	PortInterface	0..1	tref	The interface that this port requires. **Stereotypes:** isOfType

Table D.100: RPortPrototype

Class	**RPortPrototypeInSoftwareClusterDesignInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInSoftwareClusterDesignInstanceRef			
Note	**Tags:**atp.Status=draft			
Base	*ARObject*, *AbstractPortPrototypeInSoftwareClusterDesignInstanceRef*, *AtpInstanceRef*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
base	SoftwareClusterDesign	0..1	ref	**Stereotypes:** atpDerived **Tags:**atp.Status=draft
contextRootSwClusterDesignComponentPrototype	RootSwClusterDesignComponentPrototype	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10
contextSwComponentPrototype (ordered)	SwComponentPrototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetRPortPrototype	RPortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30

Table D.101: RPortPrototypeInSoftwareClusterDesignInstanceRef

Class	**RecordValueSpecification**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Specifies the values for a record.			
Base	*ARObject*, *CompositeValueSpecification*, *ValueSpecification*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
field (ordered)	ValueSpecification	*	aggr	The value for a single record field. This could also be mapped explicitly to a record element of the data type using the shortName of the ValueSpecification. But this would introduce a relationship to the data type that is too strong. As of now, it is only important that the structure of the data type matches the structure of the Value Specification indepenently of the shortNames. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime

Table D.102: RecordValueSpecification

Primitive	**Ref**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes			
Note	This primitive denotes a name based reference. For detailed syntax see the xsd.pattern. • first slash (relative or absolute reference) [optional] • Identifier [required] • a sequence of slashes and Identifiers [optional] This primitive is used by the meta-model tools to create the references. **Tags:** xml.xsd.customType=REF xml.xsd.pattern=/?[a-zA-Z][a-zA-Z0-9_]{0,127}(/[a-zA-Z][a-zA-Z0-9_]{0,127})* xml.xsd.type=string			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Identifier	0..1	attr	This attribute reflects the base to be used for this reference. **Tags:**xml.attribute=true
blueprintValue	String	0..1	attr	This represents a description that documents how the value shall be defined when deriving objects from the blueprint. **Tags:** atp.Status=draft xml.attribute=true
index	PositiveInteger	0..1	attr	This attribute supports the use case to point on specific elements in an array. This is in particular required if arrays are used to implement particular data objects. **Tags:**xml.attribute=true

Table D.103: Ref

Class	**ReferenceValueSpecification**
Package	M2::AUTOSARTemplates::CommonStructure::Constants
Note	Specifies a reference to a data prototype to be used as an initial value for a pointer in the software.
Base	*ARObject*, *ValueSpecification*

▽

△

Class	**ReferenceValueSpecification**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
referenceValue	DataPrototype	0..1	ref	The referenced data prototype.

Table D.104: ReferenceValueSpecification

Class	***Referrable*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	*ARObject*			
Subclasses	*AtpDefinition*, BswDistinguishedPartition, *BswModuleCallPoint*, BswModuleClientServerEntry, BswVariableAccess, CouplingPortTrafficClassAssignment, *CppImplementationDataTypeContextTarget*, DiagnosticDebounceAlgorithmProps, *DiagnosticEnvModeElement*, EthernetPriorityRegeneration, EventHandler, ExclusiveAreaNestingOrder, *HwDescriptionEntity*, *ImplementationProps*, LinSlaveConfigIdent, ModeTransition, *MultilanguageReferrable*, NmNetworkHandle, PduActivationRoutingGroup, PncMappingIdent, *SingleLanguageReferrable*, SoConIPduIdentifier, SocketConnectionBundle, SomeipRequiredEventGroup, TimeSyncServerConfiguration, TpConnectionIdent			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
shortName	Identifier	1	attr	This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference. **Stereotypes:** atpIdentityContributor **Tags:** xml.enforceMinMultiplicity=true xml.sequenceOffset=-100
shortNameFragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. **Tags:**xml.sequenceOffset=-90

Table D.105: Referrable

Class	**RoleBasedPortAssignment**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServiceMapping			
Note	This class specifies an assignment of a role to a particular service port (RPortPrototype or PPortPrototype) of an AtomicSwComponentType. With this assignment, the role of the service port can be mapped to a specific ServiceNeeds element, so that a tool is able to create the correct connector.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
portPrototype	PortPrototype	0..1	ref	Service PortPrototype used in the assigned role. This PortPrototype shall either belong to the same AtomicSwComponentType as the SwcInternalBehavior which owns the ServiceDependency or to the same NvBlockSwComponentType as the NvBlockDescriptor.
role	Identifier	0..1	attr	This is the role of the assigned Port in the given context. The value shall be a shortName of the Blueprint of a PortInterface as standardized in the Software Specification of the related AUTOSAR Service.

Table D.106: RoleBasedPortAssignment

Class	Sd			
Package	M2::MSR::AsamHdo::SpecialData			
Note	This class represents a primitive element in a special data group.			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
gid	NameToken	1	attr	This attributes specifies an identifier. Gid comes from the SGML/XML-Term "Generic Identifier" which is the element name in XML. The role of this attribute is the same as the name of an XML - element. **Tags:**xml.attribute=true
value	VerbatimStringPlain	1	attr	This is the value of the special data. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.typeElement=false xml.typeWrapperElement=false
xmlSpace	XmlSpaceEnum	0..1	attr	This attribute is used to signal an intention that in that element, white space should be preserved by applications. It is defined according to xml:space as declared by W3C. **Tags:** xml.attribute=true xml.attributeRef=true xml.enforceMinMultiplicity=true xml.name=space xml.nsPrefix=xml

Table D.107: Sd

Class	SecOcCryptoServiceMapping			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class has the ability to represent a crypto service mapping for the Pdu-based communication via SecOC. **Tags:**atp.recommendedPackage=CryptoServiceMappings			
Base	*ARObject*, *CryptoServiceMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
authentication	CryptoServicePrimitive	0..1	ref	This reference identifies the applicable crypto primitive for the authentication.
cryptoService Key	CryptoServiceKey	0..1	ref	This reference identifies the applicable crypto key.
cryptoService Queue	CryptoServiceQueue	0..1	ref	This reference identifies the CryptoServiceQueue the processing of this SecOcCryptoServiceMapping shall be performed in.

Table D.108: SecOcCryptoServiceMapping

Class	SecureCommunicationAuthenticationProps
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication
Note	Authentication properties used to configure SecuredIPdus.
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	**SecureCommunicationAuthenticationProps**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authInfoTx Length	PositiveInteger	0..1	attr	This attribute defines the length in bits of the authentication code to be included in the payload of the authenticated Pdu.

Table D.109: SecureCommunicationAuthenticationProps

Class	**SecureCommunicationFreshnessProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Freshness properties used to configure SecuredIPdus.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
freshness CounterSync Attempts	PositiveInteger	0..1	attr	This attribute defines the number of Freshness Counter re-synchronization attempts when a verification failed for a Secured I-PDU. If the value is zero, there will be no additional verification attempt to synchronize with a potentially better fitting Freshness Counter value. This attribute is only applicable if useFreshnessTimestamp is FALSE.
freshness TimestampTime PeriodFactor	PositiveInteger	0..1	attr	This attribute defines a factor that specifies the time period for the Freshness Timestamp. It holds a multiplication factor that specifies the concrete meaning of a Freshness Timestamp increment by one on basis of microseconds.
freshnessValue Length	PositiveInteger	0..1	attr	This attribute defines the complete length in bits of the Freshness Value. As long as the key doesn't change the counter shall not overflow. The length of the counter shall be determined based on the expected life time of the corresponding key and frequency of usage of the counter.
freshnessValue TxLength	PositiveInteger	0..1	attr	This attribute defines the length in bits of the Freshness Value to be included in the payload of the Secured I-PDU. This length is specific to the least significant bits of the complete Freshness Counter. If the attribute is 0 no Freshness Value is included in the Secured I-PDU.
useFreshness Timestamp	Boolean	0..1	attr	This attribute specifies whether the Freshness Value is generated through individual Freshness Counters or by a Timestamps. The value is set to TRUE when Timestamps are used.

Table D.110: SecureCommunicationFreshnessProps

Class	**SecureCommunicationProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	This meta-class contains configuration settings that are specific for an individual SecuredIPdu.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authData Freshness Length	PositiveInteger	0..1	attr	This attribute defines the length in bits of the authentic PDU data that is passed to the SWC that verifies and generates the Freshness.

▽

△

Class	**SecureCommunicationProps**			
authData FreshnessStart Position	PositiveInteger	0..1	attr	This value determines the start position in bits of the Authentic PDU that shall be passed on to the SWC that verifies and generates the Freshness. The bit counting is done according to TPS_SYST_01068.
authentication BuildAttempts	PositiveInteger	0..1	attr	This attribute specifies the number of authentication build attempts.
authentication Retries	PositiveInteger	1	attr	This attribute defines the additional number of authentication attempts that are to be carried out when the generation of the authentication information failed for a given SecuredIPdu. If zero is set than only one authentication attempt is done.
dataId	PositiveInteger	1	attr	This attribute defines a numerical identifier for the Secured I-PDU.
freshnessValue Id	PositiveInteger	0..1	attr	This attribute defines the Id of the Freshness Value. The Freshness Value might be a normal counter or a time value.
messageLink Length	PositiveInteger	0..1	attr	SecOC links an AuthenticIPdu and CryptographicIPdu together by repeating a specific part (Message Linker) of the AuthenticIPdu in the CryptographicIPdu. This attribute defines the length in bits of the messageLinker.
messageLink Position	PositiveInteger	0..1	attr	SecOC links an AuthenticIPdu and CryptographicIPdu together by repeating a specific part (Message Linker) of the AuthenticIPdu in the CryptographicIPdu. This attribute defines the startPosition in bits of the messageLinker.
secondary FreshnessValue Id	PositiveInteger	0..1	attr	This attribute defines the Id of the Secondary Freshness Value. The Secondary Freshness Value might be a normal counter or a time value. Please note that this attribute is for documentation only to allow the configuration of required freshness value manager and no upstream mapping is defined for it.
securedArea Length	PositiveInteger	0..1	attr	This attribute defines the length in bytes of the area within the payload Pdu which will be secured.
securedArea Offset	PositiveInteger	0..1	attr	This attribute defines the start position (offset in byte) of the area within the payload Pdu which will be secured.

Table D.111: SecureCommunicationProps

Class	**SecuredIPdu**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	If useAsCryptographicPdu is not set or set to false this IPdu contains the payload of an Authentic IPdu supplemented by additional Authentication Information (Freshness Counter and an Authenticator). If useAsCryptographicPdu is set to true this IPdu contains the Authenticator for a payload that is transported in a separate message. The separate Authentic IPdu is described by the Pdu that is referenced with the payload reference from this SecuredIPdu. **Tags:**atp.recommendedPackage=Pdus			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *IPdu*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Pdu*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authentication Props	SecureCommunication AuthenticationProps	0..1	ref	Reference to authentication properties that are valid for this SecuredIPdu.
freshnessProps	SecureCommunication FreshnessProps	0..1	ref	Reference to freshness properties that are valid for this SecuredIPdu.

▽

△

Class	**SecuredIPdu**			
payload	PduTriggering	1	ref	Reference to a Pdu that will be protected against unauthorized manipulation and replay attacks.
secure Communication Props	SecureCommunication Props	1	aggr	Specific configuration properties for this SecuredIPdu.
useAs Cryptographic IPdu	Boolean	0..1	attr	If this attribute is set to true the SecuredIPdu contains the Authentication Information for an AuthenticIPdu that is transmitted in a separate message. The AuthenticIPdu contains the original payload, i.e. the secured data. If this attribute is set to false this SecuredIPdu contains the payload of an Authentic IPdu supplemented by additional Authentication Information.
useSecuredPdu Header	SecuredPduHeader Enum	0..1	attr	This attribute defines the size of the header which is inserted into the SecuredIPdu. If this attribute is set to anything but noHeader, the SecuredIPdu contains the Secured I-PDU Header to indicate the length of the AuthenticIPdu. The AuthenticIPdu contains the original payload, i.e. the secured data.

Table D.112: SecuredIPdu

Class	**SenderReceiverInterface**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A sender/receiver interface declares a number of data elements to be sent and received. **Tags:**atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DataInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	VariableDataPrototype	*	aggr	The data elements of this SenderReceiverInterface.
invalidation Policy	InvalidationPolicy	*	aggr	InvalidationPolicy for a particular dataElement
metaDataItem Set	MetaDataItemSet	*	aggr	This aggregation defines fixed sets of meta-data items associated with dataElements of the enclosing Sender ReceiverInterface

Table D.113: SenderReceiverInterface

Class	***ServiceNeeds*** (abstract)
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
Note	This expresses the abstract needs that a Software Component or Basic Software Module has on the configuration of an AUTOSAR Service to which it will be connected. "Abstract needs" means that the model abstracts from the Configuration Parameters of the underlying Basic Software.
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*
Subclasses	BswMgrNeeds, ComMgrUserNeeds, CryptoKeyManagementNeeds, *CryptoNeeds*, CryptoServiceJob Needs, CryptoServiceNeeds, *DiagnosticCapabilityElement*, DltUserNeeds, *DoIpServiceNeeds*, EcuState MgrUserNeeds, ErrorTracerNeeds, FunctionInhibitionAvailabilityNeeds, FunctionInhibitionNeeds, Global SupervisionNeeds, HardwareTestNeeds, IdsMgrCustomTimestampNeeds, IdsMgrNeeds, IndicatorStatus Needs, J1939DcmDm19Support, J1939RmIncomingRequestServiceNeeds, J1939RmOutgoingRequest ServiceNeeds, NvBlockNeeds, SecureOnBoardCommunicationNeeds, SupervisedEntityCheckpoint Needs, SupervisedEntityNeeds, SyncTimeBaseMgrUserNeeds, V2xFacUserNeeds, V2xMUserNeeds, VendorSpecificServiceNeeds

▽

△

Class	***ServiceNeeds*** (abstract)			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table D.114: ServiceNeeds

Class	**ServiceSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	ServiceSwComponentType is used for configuring services for a given ECU. Instances of this class are only to be created in ECU Configuration phase for the specific purpose of the service configuration. **Tags:**atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table D.115: ServiceSwComponentType

Class	**SocketConnection**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ObsoleteModel			
Note	The SoAd serves as a (De)Multiplexer between different PDU sources and the TCP/IP stack. **Tags:**atp.Status=obsolete			
Base	*ARObject*, *Describable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientIpAddrFromConnectionRequest	Boolean	0..1	attr	If set to true the Server "learns" the client IP address on connection request. This means that the statically configured IP Address of the related client shall be ignored. If set to false the Server only accepts statically configured IP address, e.g. 192.168.1.2. This means that the statically configured IP Address of the Client shall be used.
clientPort	SocketAddress	0..1	ref	Client Port for TCP/UDP connection in an abstract communication sense. The client is the major requester of the communication. Please note that the client may also produce data. **Tags:**atp.Status=obsolete
clientPortFromConnectionRequest	Boolean	0..1	attr	If set to true the Server "learns" the client Port on connection request. This means that the statically configured Port of the related client shall be ignored. If set to false the Server only accepts statically configured Port. This means that the statically configured Port of the Client shall be used.
pdu	SocketConnectionIpduIdentifier	*	aggr	PDUs handed over by the PDU Router (Transmission over the Ethernet) or PDUs handed over by SoAd (Reception over Ethernet). Multiple IPdus can be transmitted over one socket connection. **Tags:**atp.Status=obsolete
pduCollectionMaxBufferSize	PositiveInteger	0..1	attr	Defines the maximum buffer size in Byte which shall be filled before a socket with Pdu collection enabled shall be transmitted to the lower layer.

▽

△

Class	**SocketConnection**			
pduCollection Timeout	TimeValue	0..1	attr	Defines the time in seconds which shall pass before a socket with Pdu collection enabled shall be transmitted to the lower layer after the first Pdu has been put into the socket buffer.
runtimeIp Address Configuration	RuntimeAddress ConfigurationEnum	0..1	attr	This attribute determines which protocol is used by the client to obtain the IP Address information. If this attribute is not set to none the value determines the service used by the client to obtain the IP Address information for the SocketConnection. If this attribute is set to none the client used the statically configured IP Address information.
runtimePort Configuration	RuntimeAddress ConfigurationEnum	0..1	attr	This attribute determines which protocol is used by the client to obtain the Port information. If this attribute is not set to none the value determines the service used by the client to obtain the Port information for the Socket Connection. If this attribute is set to none the client uses the statically configured Port information.
shortLabel	Identifier	0..1	attr	This attribute specifies an identifying shortName for the SocketConnection. It shall be unique within its context.

Table D.116: SocketConnection

Class	**SocketConnectionIpduIdentifier**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ObsoleteModel			
Note	An Identifier is required in case of one port per ECU communication where multiple Pdus are transmitted over the same connection. If only one IPdu is transmitted over the connetion this attribute can be ignored. **Tags:**atp.Status=obsolete			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
headerId	PositiveInteger	0..1	attr	If multiple Pdus are transmitted over the same connection this headerId can be used to distinguish between the different Pdus.
pduCollection PduTimeout	TimeValue	0..1	attr	Defines the timeout in seconds the PDU collection shall be transmitted at the latest after this PDU has been put into the buffer.
pduCollection Semantics	PduCollection SemanticsEnum	0..1	attr	Specifies if the referenced PduTriggering shall be collected using a queued (i.e. all PDU instances) or last-is-best (i.e. only the last PDU instance) semantics. If this attribute is not present the behavior of "queued" is assumed.
pduCollection Trigger	PduCollectionTrigger Enum	0..1	attr	Defines whether the referenced Pdu contributes to the triggering of the socket transmission if Pdu collection is enabled for this socket.
pduTriggering	PduTriggering	0..1	ref	Reference to a Pdu that is mapped to a socket connection. **Tags:**atp.Status=obsolete
routingGroup	SoAdRoutingGroup	*	ref	Reference to RoutingGroups that can be enabled or disabled. **Tags:**atp.Status=obsolete

Table D.117: SocketConnectionIpduIdentifier

Class	SwBaseType			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This meta-class represents a base type used within ECU software. **Tags:**atp.recommendedPackage=BaseTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *BaseType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table D.118: SwBaseType

Class	SwComponentPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	Role of a software component within a composition.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
type	SwComponentType	0..1	tref	Type of the instance. **Stereotypes:** isOfType

Table D.119: SwComponentPrototype

Class	***SwConnector*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	The base class for connectors between ports. Connectors have to be identifiable to allow references from the system constraint template.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AssemblySwConnector, DelegationSwConnector, PassThroughSwConnector			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
mapping	PortInterfaceMapping	0..1	ref	Reference to a PortInterfaceMapping specifying the mapping of unequal named PortInterface elements of the two different PortInterfaces typing the two PortPrototypes which are referenced by the ConnectorPrototype.

Table D.120: SwConnector

Class	<<atpVariation>> **SwDataDefProps**
Package	M2::MSR::DataDictionary::DataDefProperties
Note	This class is a collection of properties relevant for data objects under various aspects. One could consider this class as a "pattern of inheritance by aggregation". The properties can be applied to all objects of all classes in which SwDataDefProps is aggregated. Note that not all of the attributes or associated elements are useful all of the time. Hence, the process definition (e.g. expressed with an OCL or a Document Control Instance MSR-DCI) has the task of implementing limitations. SwDataDefProps covers various aspects: • Structure of the data element for calibration use cases: is it a single value, a curve, or a map, but also the recordLayouts which specify how such elements are mapped/converted to the Data ▽

▽

△

Class	<<atpVariation>> **SwDataDefProps**			
	△ Types in the programming language (or in AUTOSAR). This is mainly expressed by properties like swRecordLayout and swCalprmAxisSet • Implementation aspects, mainly expressed by swImplPolicy, swVariableAccessImplPolicy, sw AddrMethod, swPointerTagetProps, baseType, implementationDataType and additionalNative TypeQualifier • Access policy for the MCD system, mainly expressed by swCalibrationAccess • Semantics of the data element, mainly expressed by compuMethod and/or unit, dataConstr, invalidValue • Code generation policy provided by swRecordLayout **Tags:**vh.latestBindingTime=codeGenerationTime			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
additionalNative TypeQualifier	NativeDeclarationString	0..1	attr	This attribute is used to declare native qualifiers of the programming language which can neither be deduced from the baseType (e.g. because the data object describes a pointer) nor from other more abstract attributes. Examples are qualifiers like "volatile", "strict" or "enum" of the C-language. All such declarations have to be put into one string. **Tags:**xml.sequenceOffset=235
annotation	Annotation	*	aggr	This aggregation allows to add annotations (yellow pads ...) related to the current data object. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false
baseType	SwBaseType	0..1	ref	Base type associated with the containing data object. **Tags:**xml.sequenceOffset=50
compuMethod	CompuMethod	0..1	ref	Computation method associated with the semantics of this data object. **Tags:**xml.sequenceOffset=180
dataConstr	DataConstr	0..1	ref	Data constraint for this data object. **Tags:**xml.sequenceOffset=190
displayFormat	DisplayFormatString	0..1	attr	This property describes how a number is to be rendered e.g. in documents or in a measurement and calibration system. **Tags:**xml.sequenceOffset=210
display Presentation	DisplayPresentation Enum	0..1	attr	This attribute controls the presentation of the related data for measurement and calibration tools.
implementation DataType	AbstractImplementation DataType	0..1	ref	This association denotes the ImplementationDataType of a data declaration via its aggregated SwDataDefProps. It is used whenever a data declaration is not directly referring to a base type. Especially • redefinition of an ImplementationDataType via a "typedef" to another ImplementationDatatype • the target type of a pointer (see SwPointerTarget Props), if it does not refer to a base type directly ▽

▽

△

Class	<<atpVariation>> **SwDataDefProps**			
				△ • the data type of an array or record element within an ImplementationDataType, if it does not refer to a base type directly • the data type of an SwServiceArg, if it does not refer to a base type directly **Tags:**xml.sequenceOffset=215
invalidValue	ValueSpecification	0..1	aggr	Optional value to express invalidity of the actual data element. **Tags:**xml.sequenceOffset=255
stepSize	Float	0..1	attr	This attribute can be used to define a value which is added to or subtracted from the value of a DataPrototype when using up/down keys while calibrating.
swAddrMethod	SwAddrMethod	0..1	ref	Addressing method related to this data object. Via an association to the same SwAddrMethod it can be specified that several DataPrototypes shall be located in the same memory without already specifying the memory section itself. **Tags:**xml.sequenceOffset=30
swAlignment	AlignmentType	0..1	attr	The attribute describes the intended alignment of the DataPrototype. If the attribute is not defined the alignment is determined by the swBaseType size and the memory AllocationKeywordPolicy of the referenced SwAddr Method. **Tags:**xml.sequenceOffset=33
swBit Representation	SwBitRepresentation	0..1	aggr	Description of the binary representation in case of a bit variable. **Tags:**xml.sequenceOffset=60
swCalibration Access	SwCalibrationAccess Enum	0..1	attr	Specifies the read or write access by MCD tools for this data object. **Tags:**xml.sequenceOffset=70
swCalprmAxis Set	SwCalprmAxisSet	0..1	aggr	This specifies the properties of the axes in case of a curve or map etc. This is mainly applicable to calibration parameters. **Tags:**xml.sequenceOffset=90
swComparison Variable	SwVariableRefProxy	*	aggr	Variables used for comparison in an MCD process. **Tags:** xml.sequenceOffset=170 xml.typeElement=false
swData Dependency	SwDataDependency	0..1	aggr	Describes how the value of the data object has to be calculated from the value of another data object (by the MCD system). **Tags:**xml.sequenceOffset=200
swHostVariable	SwVariableRefProxy	0..1	aggr	Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects. **Tags:** xml.sequenceOffset=220 xml.typeElement=false
swImplPolicy	SwImplPolicyEnum	0..1	attr	Implementation policy for this data object. **Tags:**xml.sequenceOffset=230

▽

△

Class	<<atpVariation>> **SwDataDefProps**			
swIntended Resolution	Numerical	0..1	attr	The purpose of this element is to describe the requested quantization of data objects early on in the design process. The resolution ultimately occurs via the conversion formula present (compuMethod), which specifies the transition from the physical world to the standardized world (and vice-versa) (here, "the slope per bit" is present implicitly in the conversion formula). In the case of a development phase without a fixed conversion formula, a pre-specification can occur through swIntendedResolution. The resolution is specified in the physical domain according to the property "unit". **Tags:**xml.sequenceOffset=240
swInterpolation Method	Identifier	0..1	attr	This is a keyword identifying the mathematical method to be applied for interpolation. The keyword needs to be related to the interpolation routine which needs to be invoked. **Tags:**xml.sequenceOffset=250
swIsVirtual	Boolean	0..1	attr	This element distinguishes virtual objects. Virtual objects do not appear in the memory, their derivation is much more dependent on other objects and hence they shall have a swDataDependency . **Tags:**xml.sequenceOffset=260
swPointerTarget Props	SwPointerTargetProps	0..1	aggr	Specifies that the containing data object is a pointer to another data object. **Tags:**xml.sequenceOffset=280
swRecord Layout	SwRecordLayout	0..1	ref	Record layout for this data object. **Tags:**xml.sequenceOffset=290
swRefresh Timing	MultidimensionalTime	0..1	aggr	This element specifies the frequency in which the object involved shall be or is called or calculated. This timing can be collected from the task in which write access processes to the variable run. But this cannot be done by the MCD system. So this attribute can be used in an early phase to express the desired refresh timing and later on to specify the real refresh timing. **Tags:**xml.sequenceOffset=300
swTextProps	SwTextProps	0..1	aggr	the specific properties if the data object is a text object. **Tags:**xml.sequenceOffset=120
swValueBlock Size	Numerical	0..1	attr	This represents the size of a Value Block **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=80
swValueBlock SizeMult (ordered)	Numerical	*	attr	This attribute is used to specify the dimensions of a value block (VAL_BLK) for the case that that value block has more than one dimension. The dimensions given in this attribute are ordered such that the first entry represents the first dimension, the ▽

▽

△

Class	<<atpVariation>> **SwDataDefProps**			
				△ second entry represents the second dimension, and so on. For one-dimensional value blocks the attribute swValue BlockSize shall be used and this attribute shall not exist. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
unit	Unit	0..1	ref	Physical unit associated with the semantics of this data object. This attribute applies if no compuMethod is specified. If both units (this as well as via compuMethod) are specified the units shall be compatible. **Tags:**xml.sequenceOffset=350
valueAxisData Type	ApplicationPrimitive DataType	0..1	ref	The referenced ApplicationPrimitiveDataType represents the primitive data type of the value axis within a compound primitive (e.g. curve, map). It supersedes CompuMethod, Unit, and BaseType. **Tags:**xml.sequenceOffset=355

Table D.121: SwDataDefProps

Class	**SwPointerTargetProps**			
Package	M2::MSR::DataDictionary::DataDefProperties			
Note	This element defines, that the data object (which is specified by the aggregating element) contains a reference to another data object or to a function in the CPU code. This corresponds to a pointer in the C-language. The attributes of this element describe the category and the detailed properties of the target which is either a data description or a function signature.			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
swDataDef Props	SwDataDefProps	0..1	aggr	The properties of the target data type. **Tags:**xml.sequenceOffset=30
targetCategory	Identifier	0..1	attr	This specifies the category of the target: • In case of a data pointer, it shall specify the category of the referenced data. • In case of a function pointer, it could be used to denote the category of the referenced Bsw ModuleEntry. Since currently no categories for BswModuleEntry are defined it will be empty. **Tags:**xml.sequenceOffset=5

Table D.122: SwPointerTargetProps

Class	**SwRecordLayout**
Package	M2::MSR::DataDictionary::RecordLayout
Note	Defines how the data objects (variables, calibration parameters etc.) are to be stored in the ECU memory. As an example, this definition specifies the sequence of axis points in the ECU memory. Iterations through axis values are stored within the sub-elements swRecordLayoutGroup. **Tags:**atp.recommendedPackage=SwRecordLayouts

▽

△

Class	**SwRecordLayout**			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
swRecord LayoutGroup	SwRecordLayoutGroup	0..1	aggr	This is the top level record layout group. **Tags:** xml.roleElement=true xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

Table D.123: SwRecordLayout

Class	**SystemSignal**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	The system signal represents the communication system's view of data exchanged between SW components which reside on different ECUs. The system signals allow to represent this communication in a flattened structure, with exactly one system signal defined for each data element prototype sent and received by connected SW component instances. **Tags:**atp.recommendedPackage=SystemSignals			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicLength	Boolean	1	attr	The length of dynamic length signals is variable in run-time. Only a maximum length of such a signal is specified in the configuration (attribute length in ISignal element).
physicalProps	SwDataDefProps	0..1	aggr	Specification of the physical representation.

Table D.124: SystemSignal

Class	**TlsCryptoServiceMapping**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class has the ability to represent a crypto service mapping for the socket-based configuration of Transport Layer Security (TLS). **Tags:**atp.recommendedPackage=CryptoServiceMappings			
Base	*ARObject*, *CryptoServiceMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
keyExchange	CryptoServicePrimitive	*	ref	This reference identifies the shared(i.e. applicable for each of the aggregated cipher suites) crypto service primitive for the execution of key exchange during the handshake phase.
tlsCipherSuite	TlsCryptoCipherSuite	*	aggr	This aggregation represents the collection of supported cipher suites.

Table D.125: TlsCryptoServiceMapping

Class	**TlsPskIdentity**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This element is used to describe the pre-shared key shared during the handshake among the communication parties, to establish a TLS connection if the handshake is based on the existence of a pre-shared key.			
Base	*ARObject*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
preSharedKey	CryptoServiceKey	1	ref	This reference identifies the applicable cryptograhic key.
pskIdentity	String	1	attr	This attribute provides the key identification.
pskIdentityHint	String	0..1	attr	This attribute provides the identity hint for a pre-shared key.

Table D.126: TlsPskIdentity

Class	***TransformationProps*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	This meta-class represents a abstract base class for transformation settings.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ApSomeipTransformationProps, SOMEIPTransformationProps, UserDefinedTransformationProps			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table D.127: TransformationProps

Class	**TransmissionModeCondition**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication::Timing			
Note	Possibility to attach a condition to each signal within an I-PDU. If at least one condition evaluates to true, TRANSMISSION MODE True shall be used for this I-Pdu. In all other cases, the TRANSMISSION MODE FALSE shall be used.			
Base	*ARObject*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dataFilter	DataFilter	1	aggr	Possibilities to define conditions
iSignalInIPdu	ISignalToIPduMapping	1	ref	Reference to a signal to which a condition is attached.

Table D.128: TransmissionModeCondition

Class	**TransmissionModeDeclaration**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication::Timing
Note	AUTOSAR COM provides the possibility to define two different TRANSMISSION MODES (True and False) for each I-PDU. As TransmissionMode selector the signal content can be evaluated via transmissionModeCondition (implemented directly in the COM module) or mode conditions can be defined with the modeDrivenTrue Condition or modeDrivenFalseCondition (evaluated by BswM and invoking Com_SwitchIpduTxMode COM API). If modeDrivenTrueCondition and modeDrivenFalseCondition are defined they shall never evaluate to true both at the same time. The mixing of Transmission Mode Switch via API and signal value is not allowed.
Base	*ARObject*

△

Class	**TransmissionModeDeclaration**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
modeDriven FalseCondition	ModeDriven TransmissionMode Condition	*	aggr	Defines the trigger for the Com_SwitchIpduTxMode Transmission Mode switch. Only if all defined modeDriven FalseConditions evaluate to true (AND associated) the transmissionModeFalseTiming shall be activated. mode DrivenTrueCondition and modeDrivenFalseCondition shall never evaluate to true both at the same time.
modeDriven TrueCondition	ModeDriven TransmissionMode Condition	*	aggr	Defines the trigger for the Com_SwitchIpduTxMode Transmission Mode switch. Only if all defined modeDriven TrueConditions evaluate to true (AND associated) the transmissionModeTrueTiming shall be activated. mode DrivenTrueCondition and modeDrivenFalseCondition shall never evaluate to true both at the same time.
transmission ModeCondition	TransmissionMode Condition	*	aggr	The Transmission Mode Selector evaluates the conditions for a subset of signals and decides which transmission mode should be used. In case only one transmission mode is used there is no need for the "TransmissionMode Condition" and its sub-structure. In case the transmission mode shall be switched using the COM-API "Com_Switch IpduTxMode" there is no need for the "TransmissionMode Condition" and its sub-structure.
transmission ModeFalse Timing	TransmissionMode Timing	0..1	aggr	Timing Specification if the COM Transmission Mode is false. The Transmission Mode Selector is defined to be false, if all Conditions evaluate to false.
transmission ModeTrue Timing	TransmissionMode Timing	0..1	aggr	Timing Specification if the COM Transmission Mode is true. The Transmission Mode Selector is defined to be true, if at least one Condition evaluates to true.

Table D.129: TransmissionModeDeclaration

Enumeration	**TransportLayerProtocolEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment
Note	This enumeration allows to choose a TCP/IP transport layer protocol. **Tags:**atp.Status=draft
Literal	***Description***
tcp	Transmission control protocol **Tags:**atp.EnumerationLiteralIndex=1
udp	User datagram protocol **Tags:**atp.EnumerationLiteralIndex=0

Table D.130: TransportLayerProtocolEnum

Class	**Trigger**			
Package	M2::AUTOSARTemplates::CommonStructure::TriggerDeclaration			
Note	A trigger which is provided (i.e. released) or required (i.e. used to activate something) in the given context.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
swImplPolicy	SwImplPolicyEnum	0..1	attr	This attribute, when set to value queued, allows for a queued processing of Triggers.

▽

△

Class	Trigger			
triggerPeriod	MultidimensionalTime	0..1	aggr	Optional definition of a period in case of a periodically (time or angle) driven external trigger.

Table D.131: Trigger

Class	TriggerInterface			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A trigger interface declares a number of triggers that can be sent by an trigger source. **Tags:**atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
trigger	Trigger	*	aggr	The Trigger of this trigger interface.

Table D.132: TriggerInterface

Class	UdpNmNetworkConfiguration			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This meta-class defines the attributes for the configuration of a UDP port and UDP multicast IP address of the Nm communication on a VLAN. **Tags:**atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ipv4MulticastIp Address	Ip4AddressString	0..1	attr	Multicast IPv4 Address to which the message will be transmitted.
ipv6MulticastIp Address	Ip6AddressString	0..1	attr	Multicast IPv6 Address to which the message will be transmitted
udpPort	PositiveInteger	0..1	attr	This attribute allows to configure a udp port number that is used for reception and transmission of UdpNm messages.

Table D.133: UdpNmNetworkConfiguration

Class	Unit
Package	M2::MSR::AsamHdo::Units
Note	This is a physical measurement unit. All units that might be defined should stem from SI units. In order to convert one unit into another factor and offset are defined. For the calculation from SI-unit to the defined unit the factor (factorSiToUnit) and the offset (offsetSiTo Unit) are applied as follows: x [{unit}] := y * [{siUnit}] * factorSiToUnit [[unit]/{siUnit}] + offsetSiToUnit [{unit}] For the calculation from a unit to SI-unit the reciprocal of the factor (factorSiToUnit) and the negation of the offset (offsetSiToUnit) are applied. y {siUnit} := (x*{unit} - offsetSiToUnit [{unit}]) / (factorSiToUnit [[unit]/{siUnit}] **Tags:**atp.recommendedPackage=Units
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*

▽

△

Class	Unit			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
displayName	SingleLanguageUnitNames	0..1	aggr	This specifies how the unit shall be displayed in documents or in user interfaces of tools.The displayName corresponds to the Unit.Display in an ASAM MCD-2MC file. **Tags:**xml.sequenceOffset=20
factorSiToUnit	Float	0..1	attr	This is the factor for the conversion from SI Units to units. The inverse is used for conversion from units to SI Units. **Tags:**xml.sequenceOffset=30
offsetSiToUnit	Float	0..1	attr	This is the offset for the conversion from and to siUnits. **Tags:**xml.sequenceOffset=40
physicalDimension	PhysicalDimension	0..1	ref	This association represents the physical dimension to which the unit belongs to. Note that only values with units of the same physical dimensions might be converted. **Tags:**xml.sequenceOffset=50

Table D.134: Unit

Class	***UploadableExclusivePackageElement*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class represents an abstract base class for an uploadable package element that is not supposed to be referenced from a different software cluster. **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Subclasses	*PersistencyDeployment*, *PersistencyPortPrototypeToDeploymentMapping*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table D.135: UploadableExclusivePackageElement

Class	**UserDefinedServiceInstanceToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	This meta-class allows to map UserDefinedServiceInstances to a CommunicationConnector of a Machine. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInstanceToMachineMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *ServiceInstanceToMachineMapping*, *UploadablePackageElement*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table D.136: UserDefinedServiceInstanceToMachineMapping

Class	***ValueSpecification*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Base class for expressions leading to a value which can be used to initialize a data object.			
Base	*ARObject*			
Subclasses	*AbstractRuleBasedValueSpecification*, ApplicationValueSpecification, *CompositeValueSpecification*, ConstantReference, NotAvailableValueSpecification, NumericalValueSpecification, ReferenceValue Specification, TextValueSpecification			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
shortLabel	Identifier	0..1	attr	This can be used to identify particular value specifications for human readers, for example elements of a record type.

Table D.137: ValueSpecification

E History of Constraints and Specification Items

Please note that the lists in this chapter also include constraints and specification items that have been removed from the specification in a later version. These constraints and specification items do not appear as hyperlinks in the document.

E.1 Constraint and Specification Item History of this document according to AUTOSAR Release R17-03 (original version)

E.1.1 Created Constraints in R17-03

Number	Heading
[constr_1473]	No support for `PRPortPrototype`
[constr_1474]	`SwDataDefProps` applicable to `ImplementationDataType`s exclusive to the *AUTOSAR adaptive platform*
[constr_1475]	`ImplementationDataType` of `category` `STRING` is limited
[constr_1476]	`ImplementationDataType` of `category` `VECTOR` is limited
[constr_1477]	`ImplementationDataType` of `category` `ASSOCIATIVE_MAP` is limited
[constr_1478]	`SwDataDefProps` applicable to `ApplicationDataType`s exclusive to the *AUTOSAR adaptive platform*
[constr_1479]	No support for certain values of `ImplementationDataType.category`
[constr_1480]	Mutual existence of CompositionDataPrototypeRef.elementInImplDatatype vs. attributes of CompositionDataPrototypeRef.dataPrototype
[constr_1481]	Usage of CompositionDataPrototypeRef in the *AUTOSAR adaptive platform*
[constr_1482]	Mapping of service interfaces vs. mapping of service interface elements
[constr_1483]	Applicability of a `ServiceInterface`

▽

△

Number	Heading
[constr_1484]	Applicability of ModeDependentStartupConfig.executionDependency
[constr_1485]	No subElement for ImplementationDataType of category STRING
[constr_1486]	ImplementationDataType of category STRING and SwBaseType
[constr_1487]	Number of subElements of an ImplementationDataType of category ASSOCIATIVE_MAP
[constr_1488]	Initialization of a DataPrototype typed by an ApplicationAssocMapDataType
[constr_1489]	Uniqueness of ApplicationAssocMapValueSpecification.mapElementTuple.key
[constr_1490]	Allowed value of category for reference AdaptiveModuleInstantiation.process.executable
[constr_1491]	Reference to ApplicationError
[constr_1492]	SwComponentType referenced as Executable.rootSwComponentPrototype.applicationType
[constr_1493]	ArgumentDataPrototype referenced in the role ApplicationError.errorContext
[constr_1494]	Initial value for event
[constr_1495]	Initial value for field
[constr_1496]	DiagnosticServiceDataMapping.mappedApDataElement shall only refer to specific sub-classes of DataPrototype
[constr_1497]	Attribute optionKind set to commandLineSimpleForm
[constr_1498]	Attribute optionKind set to commandLineShortForm or commandLineLongForm
[constr_1499]	Target SwcServiceDependency of DiagnosticServiceSwMapping.mappedSwcServiceDependencyInExecutable
[constr_1500]	Target SwcServiceDependency of DiagnosticEventPortMapping.swcServiceDependencyInExecutable
[constr_1501]	Target SwcServiceDependency of DiagnosticOperationCyclePortMapping.swcServiceDependencyInExecutable
[constr_1502]	Target SwcServiceDependency of DiagnosticEnableConditionPortMapping.swcServiceDependencyInExecutable
[constr_1503]	Target SwcServiceDependency of DiagnosticStorageConditionPortMapping.swcServiceDependencyInExecutable
[constr_1504]	Number of Process.modeDependentStartupConfig that refer to the same ModeDeclaration
[constr_1505]	Number of Process.modeDependentStartupConfig that do not refer to a ModeDeclaration
[constr_1507]	PortInterfaceToDataTypeMapping is only applicable to ServiceInterface
[constr_1508]	BaseTypeDirectDefinition.nativeDeclaration shall not be set to the value enum
[constr_3320]	Aggregation of CommunicationConnector by Machine
[constr_3287]	Mandatory information of a ProvidedSomeipServiceInstance
[constr_3288]	IP configuration restriction for unicastNetworkEndpoints
[constr_3290]	Usage of ServiceInstancePortConfig defined for a ProvidedSomeipServiceInstance

▽

△

Number	Heading
[constr_3291]	SomeipServiceInstanceToMachineMapping.portConfig aggregation restriction
[constr_3293]	Mandatory information of a RequiredSomeipServiceInstance
[constr_3296]	Usage of ServiceInstancePortConfig defined for a RequiredSomeipServiceInstance
[constr_5155]	SomeipServiceInstanceToMachineMapping only supports a single Address Family
[constr_3300]	Allowed ServiceMethodDeployment.method references
[constr_3301]	Allowed ServiceEventDeployment.event references
[constr_3302]	Allowed ServiceFieldDeployment.field references
[constr_3303]	ANY not allowed for SomeipServiceInterface.serviceInterfaceVersion
[constr_3304]	Value of attribute SomeipEventGroup.eventGroupId shall be unique
[constr_3305]	Value of attribute SomeipEvent.eventId shall be unique
[constr_3306]	Value of attribute SomeipMethod.methodId shall be unique
[constr_5156]	SomeipEvent.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3308]	SomeipEvent.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3309]	SomeipMethod.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3310]	SomeipMethod.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3320]	Aggregation of CommunicationConnector by Machine
[constr_3349]	Usage of ApplicationAssocMapDataType is limited
[constr_3350]	Consistent value of category for AdaptiveAutosarApplications referencing an Executable
[constr_3351]	SOME/IP segmentation allowed for udp SomeipEvents
[constr_3352]	SOME/IP segmentation allowed for udp SomeipMethods
[constr_3353]	Restriction in usage of ApSomeipTransformationProps.sizeOfArrayLengthField
[constr_3354]	Restriction in usage of ApSomeipTransformationProps.sizeOfStructLengthField
[constr_3355]	Restriction in usage of ApSomeipTransformationProps.sizeOfUnionLengthField
[constr_3356]	Restriction in usage of ApSomeipTransformationProps.alignment
[constr_3357]	Restriction in usage of ApSomeipTransformationProps.sizeOfUnionTypeSelectorField
[constr_3358]	Usage of PortPrototype and TransportLayerIndependentInstanceId to define the same Service Instance is not allowed.
[constr_3359]	RPortPrototypeProps are related only to RPortPrototypes.
[constr_3360]	RPortPrototypeProps are related only to TransportLayerIndependentInstanceIds representing a consumer Service Instance.

▽

△

Number	Heading
[constr_3361]	Selective definition of serialization settings.
[constr_3362]	SomeipEvents aggregated by a SomeipField
[constr_3363]	SomeipMethods aggregated by a SomeipField

Table E.1: Added Constraints in original version

E.1.2 Created Specification Items in R17-03

Number	Heading
[TPS_MANI_01000]	Definition of the term `Manifest`
[TPS_MANI_01001]	Meaning of `ServiceInterface`
[TPS_MANI_01002]	Semantics of a `ServiceInterfaceMapping`
[TPS_MANI_01003]	Limitations of the applicability of `ServiceInterfaceMapping`
[TPS_MANI_01004]	Semantics of `ServiceInterface.namespace`
[TPS_MANI_01005]	The definition of the namespace of a `ServiceInterface` may follow a hierarchical pattern
[TPS_MANI_01006]	Ordered definition of `ServiceInterface.namespace`
[TPS_MANI_01007]	Service-oriented **communication** and service **discovery**
[TPS_MANI_01008]	Semantics of `AdaptiveAutosarApplication`
[TPS_MANI_01009]	Standardized values of `AdaptiveAutosarApplication.category`
[TPS_MANI_01010]	Root element for a hierarchical software-component
[TPS_MANI_01011]	Connection between application design and application deployment
[TPS_MANI_01012]	Formal modeling of application startup behavior
[TPS_MANI_01013]	Semantics of meta-class `ModeDependentStartupConfig`
[TPS_MANI_01014]	Semantics of meta-class `StartupConfigSet`
[TPS_MANI_01015]	Semantics of meta-class `StartupOption`
[TPS_MANI_01016]	Category of `ApplicationAssocMapDataType`
[TPS_MANI_01017]	Relation of startup configuration to resource groups
[TPS_MANI_01018]	`ImplementationDataType` of `category` `VECTOR`
[TPS_MANI_01019]	`Manifest` content may apply to different aspects of the *AUTOSAR adaptive platform*
[TPS_MANI_01020]	Serialization format of the *Manifest* in AUTOSAR
[TPS_MANI_01021]	Serialization format of *Manifest* content on a machine
[TPS_MANI_01022]	Concept behind `ServiceInterfaceMapping`
[TPS_MANI_01024]	Semantics of `ServiceInterfaceEventMapping`
[TPS_MANI_01025]	Semantics of `ServiceInterfaceFieldMapping`
[TPS_MANI_01026]	Semantics of `ServiceInterfaceMethodMapping`
[TPS_MANI_01027]	Semantics of `ApplicationAssocMapDataType`
[TPS_MANI_01028]	`ImplementationDataType` of `category` `ASSOCIATIVE_MAP`

▽

△

Number	Heading
[TPS_MANI_01029]	Usage of ImplementationDataType
[TPS_MANI_01030]	ImplementationDataType of category STRING
[TPS_MANI_01031]	Semantics of CompositionDataPrototypeRef
[TPS_MANI_01032]	Usage of ServiceInterfaceMapping
[TPS_MANI_01033]	Semantics of ServiceInterface.event
[TPS_MANI_01034]	Semantics of ServiceInterface.field
[TPS_MANI_01035]	Semantics of ServiceInterface.method
[TPS_MANI_01037]	Diagnostic data mapping on the *AUTOSAR adaptive platform*
[TPS_MANI_01038]	Diagnostic software mapping on the *AUTOSAR adaptive platform*
[TPS_MANI_01039]	Representation of provided service
[TPS_MANI_01040]	Representation of required service
[TPS_MANI_01041]	Startup configuration supports the definition of a launch dependency
[TPS_MANI_01042]	Definition of a linear ImplementationDataType of category VECTOR
[TPS_MANI_01043]	Definition of a rectangular ImplementationDataType of category VECTOR
[TPS_MANI_01044]	Structure of an ImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_01045]	Process.modeDependentStartupConfig that does not refer to a ModeDeclaration
[TPS_MANI_01046]	Semantics of ModeDependentStartupConfig.machineMode
[TPS_MANI_01047]	Existence of SwRecordLayout for an ApplicationPrimitiveDataType of category STRING
[TPS_MANI_01048]	Mapping of DiagnosticEvent to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01049]	Mapping of DiagnosticOperationCycle to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01050]	Mapping of DiagnosticEnableCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01051]	Mapping of DiagnosticStorageCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01052]	Semantics of RPortPrototypeProps.portInstantiationBehavior
[TPS_MANI_01053]	Usage of ComSpecs on the *AUTOSAR adaptive platform*
[TPS_MANI_01054]	Definition of the queue length of an event
[TPS_MANI_01055]	Semantics of ServiceInterface.possibleError
[TPS_MANI_01056]	Semantics of ApplicationError.errorContext
[TPS_MANI_01057]	Semantics of RPortPrototypeProps.searchIntention
[TPS_MANI_01058]	Ability to create a mapping of ApplicationErrors aggregated in the role possibleError
[TPS_MANI_01059]	Different values of optionKind within a StartupConfig.startupOption
[TPS_MANI_01060]	Use cases for the application of DiagnosticServiceDataMapping

▽

△

Number	Heading
[TPS_MANI_01061]	Requirements on scheduling
[TPS_MANI_01062]	ImplementationDataType to generate a C++ enum
[TPS_MANI_01063]	Sharing of ImplementationDataType with enumeration semantics
[TPS_MANI_03000]	Mapping of AdaptivePlatformServiceInstance to PortPrototypes
[TPS_MANI_03001]	Mapping of AdaptivePlatformServiceInstance to a Machine
[TPS_MANI_03002]	IP configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03003]	ProvidedSomeipServiceInstance Fanout
[TPS_MANI_03004]	IPv4 Multicast event destination address
[TPS_MANI_03005]	IPv4 Multicast address range
[TPS_MANI_03006]	IPv6 Multicast address range
[TPS_MANI_03007]	Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03008]	Tcp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03009]	Tcp and Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03010]	Udp Transport Protocol Configuration in case of IP-Multicast
[TPS_MANI_03011]	Server Timing configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03012]	Initial Wait Phase configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03013]	Repetition Wait Phase configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03014]	Main Phase configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03015]	TTL for Offer Service Entries
[TPS_MANI_03016]	Servers RequestResponseDelay for received FindService entries
[TPS_MANI_03017]	Server Capability Records
[TPS_MANI_03018]	Usage of SomeipProvidedEventGroup.multicastThreshold
[TPS_MANI_03019]	TTL for SubscribeEventGroupAck Entries
[TPS_MANI_03020]	Servers RequestResponseDelay for received SubscribeEventGroup entries
[TPS_MANI_03021]	Requirements on the service version from the client's point of view
[TPS_MANI_03022]	Context of RequiredSomeipServiceInstance
[TPS_MANI_03023]	Udp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03024]	Tcp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03025]	Client Timing configuration for a RequiredSomeipServiceInstance
[TPS_MANI_03026]	Initial Wait Phase configuration for a RequiredSomeipServiceInstance
[TPS_MANI_03027]	Repetition Wait Phase configuration for a RequiredSomeipServiceInstance
[TPS_MANI_03028]	TTL for Find Service Entries

▽

△

Number	Heading
[TPS_MANI_03029]	Client Capability Records
[TPS_MANI_03030]	SomeipSdClientEventGroupTimingConfig.timeToLive for SubscribeEventGroup Entries
[TPS_MANI_03031]	Clients RequestResponseDelay for received ServiceOffer entries
[TPS_MANI_03032]	Description of middleware technologies not standardized by AUTOSAR
[TPS_MANI_03035]	Content of the Machine configuration
[TPS_MANI_03036]	ServiceInterface deployment to a middleware transport layer
[TPS_MANI_03037]	Purpose of ServiceMethodDeployment
[TPS_MANI_03038]	Purpose of ServiceEventDeployment
[TPS_MANI_03039]	Purpose of ServiceFieldDeployment
[TPS_MANI_03040]	SOME/IP ServiceInterface binding
[TPS_MANI_03041]	Definition of SOME/IP EventGroups
[TPS_MANI_03042]	Definition of SOME/IP Service Version
[TPS_MANI_03043]	SOME/IP VariableDataPrototype binding
[TPS_MANI_03044]	SOME/IP ClientServerOperation binding
[TPS_MANI_03045]	UserDefined ServiceInterface binding
[TPS_MANI_03046]	User defined VariableDataPrototype binding
[TPS_MANI_03047]	User defined ClientServerOperation binding
[TPS_MANI_03048]	User defined Field binding
[TPS_MANI_03049]	Tcp and Udp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03050]	Tcp and Udp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03051]	Usage of SomeipMethod.transportProtocol
[TPS_MANI_03052]	Static IPv4 configuration
[TPS_MANI_03053]	Static IPv6 configuration
[TPS_MANI_03056]	Usage of SomeipEvent.transportProtocol
[TPS_MANI_03057]	SOME/IP Field binding
[TPS_MANI_03059]	RequiredSomeipServiceInstance.requiredServiceInstanceId
[TPS_MANI_03061]	IPv6 Multicast event destination address
[TPS_MANI_03064]	SOME/IP Service Discovery message exchange configuration
[TPS_MANI_03065]	Hardware resources of the machine
[TPS_MANI_03066]	Description of machine states
[TPS_MANI_03067]	SOME/IP segmentation of udp SomeipEvents
[TPS_MANI_03068]	SOME/IP segmentation of SomeipMethod Calls
[TPS_MANI_03069]	SOME/IP segmentation of SomeipMethod Responses
[TPS_MANI_03070]	Size of a length field for a chosen array
[TPS_MANI_03071]	Size of a length field for a chosen structure

▽

△

Number	Heading
[TPS_MANI_03072]	Size of a length field for a chosen union
[TPS_MANI_03073]	Alignment of a dynamic DataPrototype
[TPS_MANI_03074]	Size of a type selector field for a chosen union
[TPS_MANI_03075]	Byte Order of chosen DataPrototype in the serialized data stream
[TPS_MANI_03094]	Machine-specific platform configuration settings
[TPS_MANI_03095]	Implementation-specific platform configuration settings
[TPS_MANI_03096]	Machine-specific configuration settings for a generic module
[TPS_MANI_03097]	Implementation-specific configuration settings for a generic module
[TPS_MANI_03098]	Machine-specific configuration settings for the OS module
[TPS_MANI_03099]	Implementation-specific configuration settings for the OS module
[TPS_MANI_03100]	Transport layer independent TransportLayerIndependentInstanceIds
[TPS_MANI_03101]	SOME/IP serialization
[TPS_MANI_03102]	UserDefined serialization
[TPS_MANI_03103]	Default size for all array length fields
[TPS_MANI_03104]	Default size for all structure length fields
[TPS_MANI_03105]	Default size for all union length fields
[TPS_MANI_03106]	Default size for all union type selector fields
[TPS_MANI_03107]	Default alignment for all dynamic DataPrototypes
[TPS_MANI_03108]	Default Byte Order for all DataPrototypes
[TPS_MANI_03109]	TransformationProps on the level of DataPrototypes overwrites TransformationProps settings on the level of a ServiceInterface

Table E.2: Added Specification Items in original Version

E.2 Constraint and Specification Item History of this document according to AUTOSAR Release R17-10

E.2.1 Added Traceables in R17-10

Number	Heading
[TPS_MANI_01064]	Semantics of attribute method.fireAndForget
[TPS_MANI_01065]	Purpose of PersistencyKeyValueDatabaseInterface
[TPS_MANI_01067]	Purpose of PersistencyFileProxyInterface
[TPS_MANI_01068]	Semantics of PersistencyFileProxyInterface.maxNumberOfFiles
[TPS_MANI_01069]	Further qualification of properties of PortPrototypes typed by PersistencyKeyValueDatabaseInterfaces
[TPS_MANI_01073]	Semantics of PortPrototype typed by PersistencyKeyValueDatabaseInterface

▽

△

Number	Heading
[TPS_MANI_01074]	Specification of encryption of persistent data
[TPS_MANI_01075]	Specification of redundancy of persistent data
[TPS_MANI_01077]	Specification of file encryption
[TPS_MANI_01078]	Semantics of PersistencyPortPrototypeToKeyValueDatabaseMapping
[TPS_MANI_01079]	Semantics of PersistencyKeyValueDatabase
[TPS_MANI_01080]	Semantics of PersistencyFileProxyToFileMapping
[TPS_MANI_01081]	Semantics of PortPrototype typed by PersistencyFileProxyInterface
[TPS_MANI_01082]	Eligibility of DataPrototypes for the definition of optionality
[TPS_MANI_01083]	Optionality is supported for ApplicationDataType as well as ImplementationDataType
[TPS_MANI_01084]	Optionality for a DataPrototype typed by an ApplicationDataType
[TPS_MANI_01085]	Definition of optionality for a DataPrototype typed by an ImplementationDataType
[TPS_MANI_01087]	Interaction with crypto software
[TPS_MANI_01088]	Semantics of CryptoNeed
[TPS_MANI_01089]	Relation between CryptoNeed and PortPrototype
[TPS_MANI_01090]	Modeling of crypto software as a platform module
[TPS_MANI_01091]	Semantics of CryptoJob
[TPS_MANI_01092]	Mapping between CryptoNeed and CryptoJob
[TPS_MANI_01093]	Semantics of CryptoDriver
[TPS_MANI_01094]	Scope of CryptoDriver
[TPS_MANI_01095]	Semantics of CryptoKeySlot
[TPS_MANI_01096]	Semantics of the CryptoPrimitive
[TPS_MANI_01097]	Assignment of TLV data ids for data structures with optional members
[TPS_MANI_01098]	Constraints on the definition of an ImplementationDataType of category VECTOR
[TPS_MANI_01099]	Semantics of ImplementationDataTypeElementExtension
[TPS_MANI_01100]	Semantics of Allocator
[TPS_MANI_01101]	Size-constrained allocation of memory
[TPS_MANI_01102]	Specification of a namespace for an ImplementationDataType of category VECTOR
[TPS_MANI_01103]	Three-level approach to REST modeling
[TPS_MANI_01105]	Semantics of RestServiceInterface
[TPS_MANI_01106]	Specification of capabilities for the receiver of events or field notifiers
[TPS_MANI_01107]	Specification of capabilities for the sender of events or field notifiers
[TPS_MANI_01108]	Specification of capabilities for the caller of a methods or field setter/getter
[TPS_MANI_01109]	Semantics of UploadablePackageElement

▽

△

Number	Heading
[TPS_MANI_01110]	Semantics of SoftwareCluster
[TPS_MANI_01111]	Diagnostic Address of a SoftwareCluster
[TPS_MANI_01112]	Semantics of SoftwareClusterDesign
[TPS_MANI_01113]	Semantics of SoftwareClusterDesign.diagnosticAddress
[TPS_MANI_01114]	Relation of DiagnosticContributionSet to SoftwareCluster
[TPS_MANI_01115]	Specification of executable software within SoftwareCluster
[TPS_MANI_01116]	Reference to model elements included in an uploadable software package
[TPS_MANI_01117]	Semantics of SoftwareClusterDesign.intendedTargetMachine
[TPS_MANI_01118]	Relation between SoftwareClusterDesign and DiagnosticContributionSet
[TPS_MANI_01119]	Reference to model elements from SoftwareClusterDesign
[TPS_MANI_01120]	Recursive definition of RestResourceDef
[TPS_MANI_01121]	Semantics of RestResourceDef.endpoint
[TPS_MANI_01122]	Arguments to endpoints
[TPS_MANI_01123]	System Triggered Event
[TPS_MANI_01124]	Semantics of RestElementDef
[TPS_MANI_01125]	Properties of REST elements can either be primitive or have array semantics
[TPS_MANI_01126]	Definition of string properties
[TPS_MANI_01127]	Limited support for data semantics in RestAbstractNumericalPropertyDef
[TPS_MANI_01128]	Difference between RestIntegerPropertyDef and RestNumberPropertyDef
[TPS_MANI_01129]	RestObjectRef is only needed for specific implementations of REST-based communication
[TPS_MANI_01130]	Structure of a typical URI for a REST service
[TPS_MANI_01131]	Impact of nested REST resources on the structure of REST URI
[TPS_MANI_01132]	Semantics of CompositionDataPrototypeRef
[TPS_MANI_01133]	Optional element of an event
[TPS_MANI_01134]	Optional element in the context of a method
[TPS_MANI_03110]	Allowed components in system description with category category SOFTWARE_COMPONENT_SYSTEM_DESIGN_DESCRIPTION.
[TPS_MANI_03111]	Mapping between method and operation
[TPS_MANI_03112]	Mapping between an event and a dataElement
[TPS_MANI_03113]	Mapping between a field and elements of Classic Platform PortInterfaces
[TPS_MANI_03114]	Usage of AssemblySwConnectors in the System Design model
[TPS_MANI_03115]	Mapping between a fire and forget method and elements of Classic Platform PortInterfaces
[TPS_MANI_03116]	Size of a length field for a chosen string
[TPS_MANI_03117]	Default size for all string length fields

▽

△

Number	Heading
[TPS_MANI_03118]	Semantics of ServiceInterface.method with fireAndForget set to true
[TPS_MANI_03119]	Default value for the attribute fireAndForget of meta-class ClientServerOperation
[TPS_MANI_03120]	Signal-based ServiceInterface binding
[TPS_MANI_03121]	Signal-based VariableDataPrototype binding
[TPS_MANI_03122]	Signal-based Field binding
[TPS_MANI_03123]	Signal-based ClientServerOperation binding
[TPS_MANI_03124]	SignalBasedEventDeployment to ISignalTriggering mapping
[TPS_MANI_03125]	SignalBasedMethodDeployment to ISignalTriggerings mapping
[TPS_MANI_03126]	SignalBasedFieldDeployment to ISignalTriggerings mapping
[TPS_MANI_03127]	Usage of End2EndEventProtectionProps
[TPS_MANI_03128]	Usage of same dataId in case of Multi-Binding
[TPS_MANI_03129]	E2E profile
[TPS_MANI_03130]	Standardized E2EProfileConfiguration.profileName values
[TPS_MANI_03131]	Non-Standardized E2EProfileConfiguration.profileName values
[TPS_MANI_03132]	Semantics of E2E attributes in ReceiverComSpec
[TPS_MANI_03133]	Usage of ServiceInterfaceElementSecureComConfig
[TPS_MANI_03134]	Configuration of supported TLS ciphersuites
[TPS_MANI_03135]	Configuration of TLS PSK Identity
[TPS_MANI_03136]	Configuration of requirements for the TLS cryptographic job
[TPS_MANI_03137]	ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are not relevant in case of TLS communication
[TPS_MANI_03138]	SecOC Security Profile
[TPS_MANI_03139]	Standardized SecOC Security Profiles
[TPS_MANI_03140]	Non-Standardized SecOC Security Profiles
[TPS_MANI_03141]	Mapping between SecOcJobRequirement and CryptoJob
[TPS_MANI_03142]	Mapping between TlsJobRequirement and CryptoJob
[TPS_MANI_03143]	Mapping between PresharedKeyIdentity and CryptoKeySlot
[TPS_MANI_03144]	C++ language binding of ImplementationDataTypes of category STRING
[TPS_MANI_03145]	Description of a function group
[TPS_MANI_03146]	Configuration of timeouts for a selected machine state or function group state
[TPS_MANI_03147]	Mapping of a Process to a Machine
[TPS_MANI_03148]	Description of Core affinity
[TPS_MANI_03149]	Definition of a start-up timeout for a Process
[TPS_MANI_03150]	Definition of a termination timeout for a Process

▽

△

Number	Heading
[TPS_MANI_03151]	Default value for termination timeout
[TPS_MANI_03152]	Assignment of a `StateDependentStartupConfig` to a function group state
[TPS_MANI_03153]	Semantics of `ModeDependentStartupConfig.functionGroupMode`
[TPS_MANI_03500]	Definition of platform health management checkpoints
[TPS_MANI_03501]	Definition of platform health management supervised entities
[TPS_MANI_03502]	Enabling of `PlatformHealthManagementContribution` on a `Machine`
[TPS_MANI_03503]	Applicability of supervision to a specific `Process`
[TPS_MANI_03504]	Existence of `SupervisionEntity`
[TPS_MANI_03505]	Existence of `PhmCheckpoint`
[TPS_MANI_03506]	Optionality of `SupervisionEntity` and `PhmCheckpoint`
[TPS_MANI_03508]	Definition of an `AliveSupervision` for a `PhmCheckpoint`
[TPS_MANI_03509]	Definition of a `CheckpointTransition`
[TPS_MANI_03510]	Definition of `LogicalSupervision`
[TPS_MANI_03511]	Definition of `DeadlineSupervision`
[TPS_MANI_03512]	Applicability of global supervision to a specific `Process`
[TPS_MANI_03513]	Collection of `SupervisionEntity`s into a global supervision
[TPS_MANI_03514]	Expiration tolerance for `GlobalSupervisionEntity`
[TPS_MANI_03515]	Expiration tolerance for `SupervisionEntity`
[TPS_MANI_03516]	Condition evaluation for `HealthChannelSupervision`
[TPS_MANI_03517]	Condition evaluation for `HealthChannelExternalMode`
[TPS_MANI_03518]	`LogicalExpression` definition
[TPS_MANI_03519]	`Rule` definition
[TPS_MANI_03520]	Execution of `PhmActionList` with `actionListExecution=triggeredOnEvaluation`
[TPS_MANI_03521]	Execution of `PhmActionList` with `actionListExecution=triggeredOnChange`
[TPS_MANI_03522]	Definition of actions for application software
[TPS_MANI_03523]	Definition of actions for Platform Instance
[TPS_MANI_03524]	Definition of actions for Watchdog

Table E.3: Added Traceables in R17-10

E.2.2 Changed Traceables in R17-10

Number	Heading
[TPS_MANI_01004]	Semantics of `ServiceInterface.namespace`
[TPS_MANI_01006]	Ordered definition of `ServiceInterface.namespace`
[TPS_MANI_01017]	Relation of startup configuration to resource group

▽

△

Number	Heading
[TPS_MANI_01018]	ImplementationDataType of category VECTOR
[TPS_MANI_01030]	ImplementationDataType of category STRING
[TPS_MANI_03000]	Mapping of AdaptivePlatformServiceInstance to PortPrototypes
[TPS_MANI_03007]	Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03008]	Tcp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03009]	Tcp and Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03010]	Udp Transport Protocol Configuration in case of IP-Multicast
[TPS_MANI_03018]	Usage of SomeipProvidedEventGroup.multicastThreshold
[TPS_MANI_03023]	Udp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03024]	Tcp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03049]	Tcp and Udp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03101]	SOME/IP serialization
[TPS_MANI_03102]	UserDefined serialization
[TPS_MANI_03103]	Default size for all array length fields
[TPS_MANI_03104]	Default size for all structure length fields
[TPS_MANI_03105]	Default size for all union length fields
[TPS_MANI_03106]	Default size for all union type selector fields
[TPS_MANI_03107]	Default alignment for all dynamic DataPrototypes
[TPS_MANI_03108]	Default Byte Order for all DataPrototypes
[TPS_MANI_03109]	TransformationProps on the level of DataPrototypes overwrites TransformationProps settings on the level of a ServiceInterface

Table E.4: Changed Traceables in R17-10

E.2.3 Deleted Traceables in R17-10

Number	Heading
[TPS_MANI_03100]	Transport layer independent TransportLayerIndependentInstanceIds

Table E.5: Deleted Traceables in R17-10

E.2.4 Added Constraints in R17-10

Number	Heading
[constr_1522]	Semantics of ClientServerOperation.possibleError
[constr_1524]	Standardized values of PersistencyFileProxyInterface.category
[constr_1525]	Standardized values of PersistencyFile.category
[constr_1526]	Values of PersistencyFileArray.file.category
[constr_1527]	ImplementationDataTypeElement finally referenced as the target element in the context of an ImplementationDataTypeElementInAutosarDataPrototypeRef
[constr_1528]	Definition of optionality for multiple DataPrototypes typed by the same AutosarDataType
[constr_1529]	Standardized values of CryptoNeed.category
[constr_1530]	Standardized values of CryptoPrimitive.algorithmFamily and CryptoKeySlot.algorithmFamily
[constr_1531]	Standardized values of CryptoPrimitive.algorithmMode
[constr_1532]	Consistent assignment of TLV data ids to data structures with optional members
[constr_1533]	Applicability of ImplementationDataTypeElementExtension
[constr_1534]	Existence of DiagnosticSoftwareClusterProps
[constr_1535]	Existence of DiagnosticSoftwareClusterProps in the context of a DiagnosticContributionSet
[constr_1536]	Definition of SoftwareCluster applies for a single Machine
[constr_1537]	Consistent assignment of TLV data ids to arguments of a given ClientServerOperation
[constr_1542]	No nested definition of SoftwareCluster
[constr_1543]	Only one physical address per SoftwareCluster
[constr_3366]	System category for a system description with Adaptive Platform components
[constr_3367]	FieldMapping.notifierDataElement reference
[constr_3368]	FieldMapping.getterOperation reference
[constr_3369]	FieldMapping.setterOperation reference
[constr_3370]	InterfaceMapping shall map all elements of a single ServiceInterface
[constr_3371]	Mutually exclusive existence of FireAndForgetMapping.dataElement reference and FireAndForgetMapping.trigger reference
[constr_3372]	Restriction in usage of ApSomeipTransformationProps.sizeOfStringLengthField
[constr_3374]	method with attribute fireAndForget set to true shall not have any inout or out arguments
[constr_3375]	method with attribute fireAndForget set to true shall not reference an ApplicationError
[constr_3376]	FireAndForgetMapping shall reference only fire and forget methods
[constr_3377]	Restriction of ISignalTriggering references in SignalBasedFieldToISignalTriggeringMapping
[constr_3380]	End2EndEventProtectionProps shall not reference an event and a notifier at the same time

▽

△

Number	Heading
[constr_3387]	Compatibility of PortPrototypes of different ServiceInterfaces
[constr_3388]	Compatibility of events
[constr_3389]	Compatibility of methods
[constr_3390]	Compatibility of fields
[constr_3391]	ServiceInterfaceElementSecureComConfig references to ServiceInterfaceDeployment elements
[constr_3392]	ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are mandatory in case of SecOC communication
[constr_3393]	Usage of shallRunOn and shallNotRunOn references
[constr_3394]	Default value for start-up timeout on the Machine is not configurable
[constr_3395]	TransformationPropsToServiceInterfaceElementMapping is restricted to one single ServiceInterface
[constr_3396]	Number of Process.modeDependentStartupConfig that refer to the same functionGroupMode
[constr_3397]	ModeDependentStartupConfig that refers to a functionGroupMode and to a machineMode
[constr_3398]	ModeDependentStartupConfig that refers to function group modes of different function groups
[constr_3527]	LogicalExpression referenced by one PhmRule

Table E.6: Added Constraints in R17-10

E.2.5 Changed Constraints in R17-10

Number	Heading
[constr_1486]	ImplementationDataType of category STRING and SwBaseType
[constr_1490]	Allowed value of category for reference ProcessToMachineMapping.process.executable
[constr_3290]	Transport Protocol attributes defined for a ProvidedSomeipServiceInstance
[constr_3296]	Transport Protocol attributes defined for a RequiredSomeipServiceInstance
[constr_3307]	SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3308]	SomeipEventDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3309]	SomeipMethodDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3310]	SomeipMethodDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3361]	Selective definition of serialization settings

Table E.7: Changed Constraints in R17-10

E.2.6 Deleted Constraints in R17-10

Number	Heading
[constr_3291]	SomeipServiceInstanceToMachineMapping.portConfig aggregation restriction
[constr_3358]	Usage of PortPrototype and TransportLayerIndependentInstanceId to define the same Service Instance is not allowed
[constr_3360]	RPortPrototypeProps are related only to TransportLayerIndependentInstanceIds representing a consumer Service Instance

Table E.8: Deleted Constraints in R17-10

E.3 Constraint and Specification Item History of this document according to AUTOSAR Release R18-03

E.3.1 Added Traceables in R18-03

Number	Heading
[TPS_MANI_01135]	Semantics of PersistencyKeyValueDatabaseInterface.dataTypeForSerialization
[TPS_MANI_01136]	AutosarDataPrototype is the target of the CompositionDataPrototypeRef
[TPS_MANI_01137]	Applicable use cases for CompositionDataPrototypeRef
[TPS_MANI_01138]	Semantics of PersistencyKeyValueDatabaseInterface.dataElement
[TPS_MANI_01139]	Semantics of PersistencyKeyValueDatabaseInterface.updateStrategy
[TPS_MANI_01140]	Semantics of PersistencyDataElement.updateStrategy
[TPS_MANI_01141]	Semantics of PersistencyFileProxyInterface.updateStrategy
[TPS_MANI_01142]	Semantics of PersistencyFileProxy
[TPS_MANI_01143]	Semantics of PersistencyFileProxy.updateStrategy
[TPS_MANI_01144]	Semantics of PersistencyKeyValuePair
[TPS_MANI_01146]	Initial value for PersistencyKeyValuePair
[TPS_MANI_01147]	Semantics of PersistencyKeyValueDatabase.updateStrategy
[TPS_MANI_01148]	Semantics of PersistencyKeyValuePair.updateStrategy
[TPS_MANI_01149]	Semantics of PersistencyFileArray.file
[TPS_MANI_01150]	Semantics of PersistencyFileArray
[TPS_MANI_01151]	Semantics of PersistencyFileArray.updateStrategy
[TPS_MANI_01152]	Semantics of PersistencyFile.updateStrategy
[TPS_MANI_01154]	PersistencyFileArray.updateStrategy overrides PersistencyFileProxyInterface.updateStrategy
[TPS_MANI_01155]	PersistencyKeyValueDatabase.updateStrategy overrides PersistencyKeyValueDatabaseInterface.updateStrategy

△

Number	Heading
[TPS_MANI_01156]	PersistencyKeyValuePair.updateStrategy overrides PersistencyKeyValueDatabase.updateStrategy
[TPS_MANI_01157]	Semantics of updateStrategy on collection level
[TPS_MANI_01158]	PersistencyFile.updateStrategy overrides PersistencyFileArray.updateStrategy
[TPS_MANI_01159]	Semantics of updateStrategy on element level
[TPS_MANI_01160]	Definition of initial value for PersistencyDataElement
[TPS_MANI_01161]	Impact of values of category on the semantics of SoftwareClusterDesign
[TPS_MANI_01162]	Semantics of SoftwareClusterDesign.dependsOn
[TPS_MANI_01163]	Impact of values of category on the semantics of SoftwareCluster
[TPS_MANI_01164]	Semantics of SoftwareCluster.dependsOn
[TPS_MANI_01165]	Standardized value of UserDefinedServiceInterfaceDeployment.category
[TPS_MANI_01166]	Semantics of CppImplementationDataType
[TPS_MANI_01167]	AbstractImplementationDataType
[TPS_MANI_01168]	Specification of a namespace for a CppImplementationDataType
[TPS_MANI_01169]	Support for template data types
[TPS_MANI_01170]	Semantics of CppTemplateArgument.isVariadicTemplate
[TPS_MANI_01171]	Modeling of structured data types
[TPS_MANI_01172]	Description of type references in the scope of CppImplementationDataType
[TPS_MANI_01173]	Description of type references in the scope of CppImplementationDataTypeElement
[TPS_MANI_01174]	Semantics of reference in the role CppTemplateArgument.templateType
[TPS_MANI_01175]	Semantics of reference in the role CppTemplateArgument.allocator
[TPS_MANI_01176]	Standardized value for attribute CppImplementationDataType.typeEmitter
[TPS_MANI_01177]	Semantics of CppImplementationDataType.typeEmitter
[TPS_MANI_01178]	Semantics of RestHttpPortPrototypeMapping.acceptsEncoding
[TPS_MANI_01179]	Semantics of PersistencyFileProxy.contentUri/PersistencyFile.contentUri vs. PersistencyFileArray.uri and PersistencyFileProxy.fileName/PersistencyFile.fileName
[TPS_MANI_01180]	Collection of data types that requires serialization support
[TPS_MANI_01181]	Use cases for the application of DiagnosticServiceSwMapping
[TPS_MANI_01182]	PersistencyKeyValuePair.updateStrategy overrides PersistencyDataElement.updateStrategy
[TPS_MANI_01183]	PersistencyFile.updateStrategy overrides PersistencyFileProxy.updateStrategy
[TPS_MANI_03154]	ProvidedSomeipServiceInstance related configuration settings for events

▽

△

Number	Heading
[TPS_MANI_03155]	ProvidedSomeipServiceInstance related configuration settings for methods
[TPS_MANI_03156]	RequiredSomeipServiceInstance related configuration settings for methods
[TPS_MANI_03157]	Enabling of data accumulation for upd data transmission
[TPS_MANI_03158]	Configuration of a data accumulation on a ProvidedServiceInstance for transmission over udp
[TPS_MANI_03159]	Configuration of a data accumulation on a RequiredSomeipServiceInstance for transmission over udp
[TPS_MANI_03160]	Log and Trace configuration options in the Application Manifest
[TPS_MANI_03161]	Log and Trace configuration options in the Service Instance Manifest
[TPS_MANI_03162]	Machine-specific configuration settings for the Log and Trace functional cluster
[TPS_MANI_03163]	Network configuration for Log and Trace messages
[TPS_MANI_03164]	Machine-specific configuration settings for DoIP
[TPS_MANI_03165]	Network configuration for DoIP
[TPS_MANI_03166]	Machine-specific configuration settings for NM module
[TPS_MANI_03167]	Network configuration for Nm
[TPS_MANI_03168]	Configuration of the SOME/IP load balancing option
[TPS_MANI_03169]	CppImplementationDataType with fixed size array semantics
[TPS_MANI_03170]	CppImplementationDataType of category ARRAY
[TPS_MANI_03171]	Value type of a CppImplementationDataType of category ARRAY
[TPS_MANI_03172]	Size of a CppImplementationDataType of category ARRAY
[TPS_MANI_03173]	multidimensional Array
[TPS_MANI_03174]	CppImplementationDataType with variable size array semantics
[TPS_MANI_03175]	CppImplementationDataType of category VECTOR
[TPS_MANI_03176]	Value type of a CppImplementationDataType of category VECTOR
[TPS_MANI_03177]	multidimensional Vector
[TPS_MANI_03178]	CppImplementationDataType of category STRING
[TPS_MANI_03179]	C++ language binding of CppImplementationDataTypes of category STRING
[TPS_MANI_03180]	Definition of Structures
[TPS_MANI_03181]	Definition of members in CppImplementationDataType of category STRUCTURE
[TPS_MANI_03182]	Definition of members in CppImplementationDataTypeElement of category STRUCTURE
[TPS_MANI_03183]	CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03184]	CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03185]	Structure of an CppImplementationDataType of category ASSOCIATIVE_MAP

▽

△

Number	Heading
[TPS_MANI_03186]	Usage of `arraySize` in case of a Vector
[TPS_MANI_03187]	Definition of enumeration types
[TPS_MANI_03188]	Usage of an Allocator for a `CppImplementationDataType` of `category STRING`
[TPS_MANI_03189]	Definition of `CppImplementationDataType` of `category VARIANT`
[TPS_MANI_03190]	`CppImplementationDataType` of `category VARIANT`
[TPS_MANI_03191]	Definition of type alternatives stored in a `VARIANT`
[TPS_MANI_03192]	`CppImplementationDataType` of `category VALUE`
[TPS_MANI_03193]	`CppImplementationDataType` or `CppImplementationDataTypeElement` of `category TYPE_REFERENCE`
[TPS_MANI_03194]	Function Group State
[TPS_MANI_03195]	Off state in Function Group
[TPS_MANI_03196]	Semantics of `CppImplementationDataTypeElementQualifier.anonymous` attribute
[TPS_MANI_03525]	DDS ServiceInterface binding
[TPS_MANI_03526]	DDS `VariableDataPrototype` binding
[TPS_MANI_03527]	Definition of `ProvidedDdsServiceInstance`
[TPS_MANI_03528]	Definition of `ProvidedDdsEventQosProps`
[TPS_MANI_03529]	Definition of `RequiredDdsServiceInstance`
[TPS_MANI_03530]	Definition of `RequiredDdsEventQosProps`
[TPS_MANI_03531]	`qosProfile` of `ProvidedDdsEventQosProps` is optional
[TPS_MANI_03532]	`qosProfile` of `RequiredDdsEventQosProps` is optional
[TPS_MANI_03533]	`DdsServiceInstanceToMachineMapping`
[TPS_MANI_03534]	Definition of Platform Health Management Health Channel
[TPS_MANI_03535]	Definition of Time Synchronization interaction
[TPS_MANI_03536]	Time Synchronization interaction in a master role
[TPS_MANI_03537]	Time Synchronization interaction in a slave role
[TPS_MANI_03538]	Time Synchronization interaction with a local Time Base
[TPS_MANI_03539]	Definition of Time Bases
[TPS_MANI_03540]	Definition of `PureLocalTimeBase`
[TPS_MANI_03541]	Definition of `SynchronizedSlaveTimeBase`
[TPS_MANI_03542]	Definition of `SynchronizedMasterTimeBase`
[TPS_MANI_03543]	Definition of time sync correction attributes
[TPS_MANI_03544]	Definition of `PlatformHealthManagementContribution`
[TPS_MANI_03545]	Existence of `HealthChannelExternalStatus`
[TPS_MANI_03546]	Definition of reported health status `RPortPrototype`
[TPS_MANI_03547]	Definition of *offset* time domains
[TPS_MANI_03548]	Definition of `TimeSyncPortPrototypeToTimeBaseMapping`

▽

△

Number	Heading
[TPS_MANI_03549]	Usage of RPortPrototype for the interaction with Time Synchronization
[TPS_MANI_03550]	Usage of RPortPrototype for the interaction with Platform Health Management
[TPS_MANI_03551]	Definition of Time Base kind
[TPS_MANI_03552]	Supervision cycle for GlobalSupervision

Table E.9: Added Traceables in R18-03

E.3.2 Changed Traceables in R18-03

Number	Heading
[TPS_MANI_01006]	Ordered definition of ServiceInterface.namespace
[TPS_MANI_01008]	Semantics of ExecutableGroup
[TPS_MANI_01009]	Standardized values of ExecutableGroup.category
[TPS_MANI_01013]	Semantics of meta-class ModeDependentStartupConfig
[TPS_MANI_01017]	Relation of startup configuration to resource group
[TPS_MANI_01041]	Startup configuration supports the definition of a launch sequence dependency
[TPS_MANI_01042]	Definition of a linear ImplementationDataType of category VECTOR
[TPS_MANI_01044]	Structure of an ImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_01060]	Use cases for the application of DiagnosticServiceDataMapping
[TPS_MANI_01068]	Semantics of PersistencyFileProxyInterface.maxNumberOfFiles
[TPS_MANI_01069]	Further qualification of properties of PortPrototypes typed by PersistencyKeyValueDatabaseInterfaces
[TPS_MANI_01075]	Specification of redundancy of persistent data
[TPS_MANI_01078]	Semantics of PersistencyPortPrototypeToKeyValueDatabaseMapping
[TPS_MANI_01080]	Semantics of PersistencyPortPrototypeToFileArrayMapping
[TPS_MANI_01097]	Assignment of TLV data ids for data structures with optional members
[TPS_MANI_01100]	Semantics of Allocator
[TPS_MANI_01109]	Semantics of UploadablePackageElement
[TPS_MANI_01112]	Semantics of SoftwareClusterDesign
[TPS_MANI_01113]	Semantics of SoftwareClusterDesign.diagnosticAddress
[TPS_MANI_01116]	Reference to model elements included in an uploadable software package
[TPS_MANI_01117]	Semantics of SoftwareClusterDesign.intendedTargetMachine
[TPS_MANI_01118]	Relation between SoftwareClusterDesign and DiagnosticContributionSet
[TPS_MANI_01119]	Reference to model elements from SoftwareClusterDesign
[TPS_MANI_01133]	Optional element of an event

▽

△

Number	Heading
[TPS_MANI_01134]	Optional element in the context of a `method`
[TPS_MANI_03001]	Mapping of `AdaptivePlatformServiceInstance` to a `MachineDesign`
[TPS_MANI_03002]	IP configuration for a `ProvidedSomeipServiceInstance`
[TPS_MANI_03003]	`ProvidedSomeipServiceInstance` Fanout
[TPS_MANI_03022]	Context of `RequiredSomeipServiceInstance`
[TPS_MANI_03110]	Allowed components in system description with `category` SYSTEM_DESIGN_DESCRIPTION.
[TPS_MANI_03114]	Usage of `AssemblySwConnector`s in the System Design model
[TPS_MANI_03145]	Description of a function group
[TPS_MANI_03152]	Assignment of a `ModeDependentStartupConfig` to a function group state
[TPS_MANI_03153]	Semantics of `ModeDependentStartupConfig.functionGroupMode`
[TPS_MANI_03500]	Definition of Platform Health Management Supervision and Checkpoints
[TPS_MANI_03503]	Applicability of supervision to a specific `Process`
[TPS_MANI_03505]	Existence of `SupervisionCheckpoint`
[TPS_MANI_03506]	Optionality of `SupervisionCheckpoint`
[TPS_MANI_03508]	Definition of an `AliveSupervision` for a `SupervisionCheckpoint`
[TPS_MANI_03509]	Definition of a `CheckpointTransition`
[TPS_MANI_03510]	Definition of `LogicalSupervision`
[TPS_MANI_03512]	Applicability of global supervision to a specific `Process`
[TPS_MANI_03513]	Collection of `LocalSupervision`s into a global supervision
[TPS_MANI_03514]	Expiration tolerance for `GlobalSupervision`
[TPS_MANI_03515]	Expiration tolerance for `LocalSupervision`
[TPS_MANI_03516]	Condition evaluation for `HealthChannelSupervision`
[TPS_MANI_03517]	Condition evaluation for `HealthChannelExternalStatus`

Table E.10: Changed Traceables in R18-03

E.3.3 Deleted Traceables in R18-03

Number	Heading
[TPS_MANI_01031]	Semantics of `CompositionDataPrototypeRef`
[TPS_MANI_01045]	`Process`.`modeDependentStartupConfig` that does not refer to a `ModeDeclaration`
[TPS_MANI_01132]	Semantics of `CompositionDataPrototypeRef`
[TPS_MANI_03019]	TTL for `SubscribeEventGroupAck` Entries
[TPS_MANI_03501]	Definition of platform health management supervised entities
[TPS_MANI_03504]	Existence of `SupervisionEntity`

Table E.11: Deleted Traceables in R18-03

E.3.4 Added Constraints in R18-03

Number	Heading
[constr_1546]	Existence of attributes of ServiceInterfaceSubElement
[constr_1547]	Reference from ImplementationDataTypeExtension to ImplementationDataType
[constr_1548]	Reference from ImplementationDataTypeElementExtension to ImplementationDataTypeElement
[constr_1549]	Value of ProcessorCore.coreId
[constr_1550]	Reference from Process to ProcessDesign
[constr_1551]	Existence of CompositionDataPrototypeRef.dataPrototype vs. CompositionDataPrototypeRef.elementInImplDatatype
[constr_1553]	Restriction for ProcessToMachineMapping
[constr_1554]	Restriction regarding PersistencyKeyValuePair.initValue
[constr_1555]	Restriction applicable for PersistencyPortPrototypeToKeyValueDatabaseMapping.portPrototype
[constr_1556]	Restriction applicable for PersistencyPortPrototypeToFileArrayMapping.portPrototype
[constr_1557]	Standardized values of SoftwareClusterDesign.category and SoftwareCluster.category
[constr_1558]	Existence of SoftwareClusterDesign.diagnosticAddress
[constr_1559]	Existence of SoftwareClusterDesign.subSoftwareCluster
[constr_1560]	Usage of SoftwareClusterDesign.requiredARElement
[constr_1561]	Existence of SoftwareClusterDesign.subSoftwareCluster and SoftwareClusterDesign.dependsOn.dependentSoftwareClusterDesign
[constr_1562]	Existence of SoftwareClusterDesign.diagnosticContribution
[constr_1563]	Standardized values of SoftwareClusterDesign.category and SoftwareCluster.category
[constr_1564]	Existence of SoftwareCluster.diagnosticAddress
[constr_1565]	Existence of SoftwareCluster.subSoftwareCluster
[constr_1566]	Usage of SoftwareCluster.containedARElement
[constr_1567]	Existence of SoftwareCluster.subSoftwareCluster and SoftwareCluster.dependsOn.dependentSoftwareCluster
[constr_1568]	Existence of SoftwareCluster.diagnosticExtract
[constr_1569]	Restriction for the scope of RestHttpPortPrototypeMapping.acceptsEncoding
[constr_1570]	Restriction for UserDefinedServiceInterfaceDeployment of category SERVICE_INTERFACE_DEPLOYMENT_IPC
[constr_1571]	CppImplementationDataType is limited
[constr_1572]	Usage of SwDataDefProps.implementationDataType within a CppImplementationDataType
[constr_1573]	CppTemplateArgument.isVariadicTemplate is set to True
[constr_1574]	Number of CppTemplateArguments with isVariadicTemplate set to True
[constr_1575]	Position of CppTemplateArgument with isVariadicTemplate set to True

▽

△

Number	Heading
[constr_1576]	Existence of CppTemplateArgument.templateType vs. CppTemplateArgument.allocator
[constr_1577]	Specification of a nativeDeclaration for a CppImplementationDataType
[constr_1578]	applicable data categories
[constr_1579]	SwDataDefProps applicable to CppImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*
[constr_1580]	Restriction for the usage of RestHttpPortPrototypeMapping.acceptsEncoding
[constr_1581]	Value of fileProxy.fileName
[constr_1582]	PersistencyKeyValuePair.valueDataType shall match to ImplementationDataType for the corresponding PersistencyDataElement
[constr_1585]	Standardized values of attribute DiagnosticServiceSwMapping.category
[constr_1586]	DiagnosticServiceSwMapping.category set to DATA_ELEMENT
[constr_1587]	DiagnosticServiceSwMapping.category set to DATA_IDENTIFIER
[constr_1588]	DiagnosticServiceSwMapping.category set to GENERIC_UDS_SERVICE
[constr_1589]	Value of file.fileName
[constr_3408]	Value range of SomeipEventDeployment.eventId
[constr_3409]	Value range of SomeipMethodDeployment.methodId
[constr_3410]	Value range of SomeipServiceInterfaceDeployment.serviceInterfaceId
[constr_3411]	eventMulticastUdpPort, ipv4MulticastIpAddress and ipv6MulticastIpAddress not relevant for RequiredSomeipServiceInstances
[constr_3412]	OsModuleInstantiation shall have at least one ResourceGroup
[constr_3413]	ModeDependentStartupConfig of a Process is mapped to exactly one ResourceGroup
[constr_3414]	Allowed usage of EthernetNetworkConfiguration attributes
[constr_3415]	Value range of loadBalancingPriority
[constr_3416]	Value range of loadBalancingWeight
[constr_3417]	UserDefinedEventDeployments aggregated by a UserDefinedFieldDeployment
[constr_3418]	UserDefinedMethodDeployments aggregated by a UserDefinedFieldDeployment
[constr_3419]	Allowed usage of EthernetNetworkConfiguration attributes
[constr_3420]	System category for a design description that has one single Adaptive Machine in scope
[constr_3421]	Fibex elements applicable for a MACHINE_DESIGN_EXTRACT
[constr_3422]	CppImplementationDataType of category STRING and SwBaseType
[constr_3423]	ModeDependentStartupConfig of a Process shall reference a functionGroupMode or machineMode
[constr_3424]	ModeDependentStartupConfig shall never reference the functionGroupMode Off
[constr_3425]	Restriction of DoIpInstantiations on a Machine

▽

△

Number	Heading
[constr_3426]	The logTraceFilePath is mandatory in case that logTraceLogMode is set to file
[constr_3427]	The logTraceFilePath is only relevant if logTraceLogMode is set to file
[constr_3428]	Structure shall own at least one element
[constr_3429]	No allocator usage for CppImplementationDataTypes of category VARIANT
[constr_3432]	Allowed subElements for Structures
[constr_3433]	Aggregation of templateArguments for a ARRAY
[constr_3434]	Aggregation of templateArguments for a VECTOR
[constr_3528]	Value range of domainId
[constr_3529]	Value range of serviceInstanceId
[constr_3530]	Mandatory definition of checkpointId
[constr_3531]	Mandatory definition of healthChannelId
[constr_3532]	Mandatory definition of statusId
[constr_3536]	Mandatory definition of supervisedEntityId

Table E.12: Added Constraints in R18-03

E.3.5 Changed Constraints in R18-03

Number	Heading
[constr_1484]	Applicability of ModeDependentStartupConfig.executionDependency
[constr_1507]	PortInterfaceToDataTypeMapping is only applicable to ServiceInterface
[constr_1532]	Consistent assignment of TLV data ids to data structures with optional members
[constr_1537]	Consistent assignment of TLV data ids to arguments of a given ClientServerOperation
[constr_3307]	SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3308]	SomeipEventDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3309]	SomeipMethodDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3310]	SomeipMethodDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3320]	Aggregation of CommunicationConnector by MachineDesign
[constr_3350]	Consistent value of category for ExecutableGroups referencing an Executable
[constr_3366]	System category for a system design description with Adaptive Platform and Classic Platform content

Table E.13: Changed Constraints in R18-03

E.3.6 Deleted Constraints in R18-03

Number	Heading
[constr_1480]	Mutual existence of CompositionDataPrototypeRef.elementInImplDatatype vs. attributes of CompositionDataPrototypeRef.dataPrototype
[constr_1505]	Number of Process.modeDependentStartupConfig that do not refer to a ModeDeclaration
[constr_1525]	Standardized values of PersistencyFile.category
[constr_1526]	Values of PersistencyFileArray.file.category
[constr_1533]	Applicability of ImplementationDataTypeElementExtension

Table E.14: Deleted Constraints in R18-03

E.4 Constraint and Specification Item History of this document according to AUTOSAR Release R18-10

E.4.1 Added Traceables in R18-10

Number	Heading
[TPS_MANI_01184]	Definition of optional elements on the level of ApplicationDataType
[TPS_MANI_01185]	Definition of optional elements on the level of CppImplementationDataType
[TPS_MANI_01186]	Definition of the applicable wire type
[TPS_MANI_01187]	Matching pairs of PersistencyFileProxy and PersistencyFile
[TPS_MANI_01188]	Semantics of attribute schedulingPriority
[TPS_MANI_01189]	Software Cluster and DiagnosticContributionSet.category
[TPS_MANI_01190]	Semantics of ApApplicationError
[TPS_MANI_01191]	Modeling of possible errors
[TPS_MANI_01192]	Semantics of ApApplicationErrorDomain
[TPS_MANI_01193]	Combination of ModeDependentStartupConfig.machineMode and ModeDependentStartupConfig.functionGroupMode
[TPS_MANI_01194]	Semantics of PersistencyKeyValueDatabaseInterface.minimumSustainedSize
[TPS_MANI_01195]	Semantics of PersistencyFileProxyInterface.minimumSustainedSize
[TPS_MANI_01196]	Semantics of PersistencyKeyValueDatabase.minimumSustainedSize
[TPS_MANI_01197]	Semantics of PersistencyKeyValueDatabase.maximumAllowedSize
[TPS_MANI_01198]	Semantics of ApApplicationErrorSet
[TPS_MANI_01199]	Semantics of DeterministicClientResourceNeeds
[TPS_MANI_01200]	Semantics of meta-class DeterministicClientResource
[TPS_MANI_01201]	Standardized values for attribute CppTemplateArgument.category
[TPS_MANI_01202]	Semantics of reference SoftwareCluster.moduleInstantiation
[TPS_MANI_01203]	Semantics of DeterministicClient

△

Number	Heading
[TPS_MANI_01204]	Specification of redundancy of persistent data
[TPS_MANI_01205]	Semantics of meta-class PersistencyDeployment
[TPS_MANI_01206]	Modeling of redundancy in the context of PersistencyDeployment
[TPS_MANI_01207]	Standardized values of attribute PersistencyRedundancyCrc.algorithmFamily
[TPS_MANI_01208]	Definition of envionment variables in the scope of a Machine
[TPS_MANI_01209]	Definition of envionment variables in process scope
[TPS_MANI_01210]	Default encoding for all DataPrototypes typed by CppImplementationDataType of category STRING
[TPS_MANI_03197]	Semantics of StdCppImplementationDataType
[TPS_MANI_03198]	Semantics of CustomCppImplementationDataType
[TPS_MANI_03199]	Endpoint protection by SecureComProps
[TPS_MANI_03200]	SecureComProps for udp, tcp and multicast communication
[TPS_MANI_03201]	Semantics of CppTemplateArgument.inplace attribute
[TPS_MANI_03202]	Definition of bitfield types
[TPS_MANI_03203]	Configuration of IPsec
[TPS_MANI_03204]	Definition of IPSecRules
[TPS_MANI_03205]	IPsec connection type
[TPS_MANI_03206]	IPsec AH and ESP protocol configuration
[TPS_MANI_03207]	IPsec Internet Key Exchange protocol configuration
[TPS_MANI_03208]	Protection of AdaptivePlatformServiceInstance by IPsec
[TPS_MANI_03209]	The meaning of MachineDesign.accessControl
[TPS_MANI_03210]	Specification of event specific communication attributes
[TPS_MANI_03211]	Specification of field specific communication attributes
[TPS_MANI_03212]	Specification of initial value for a field
[TPS_MANI_03213]	Semantics of meta-class TlsSecureComProps
[TPS_MANI_03214]	Existence of TlsCryptoCipherSuite.keyExchange vs. TlsSecureComProps.keyExchange
[TPS_MANI_03215]	Semantics of CryptoServiceCertificate
[TPS_MANI_03216]	Existence of TlsCryptoCipherSuite.certificate in the *client* role
[TPS_MANI_03217]	On-the-wire encoding for a chosen string
[TPS_MANI_03218]	Default value for the attribute tcpInitialInactivityTime of meta-class DoIpNetworkConfiguration
[TPS_MANI_03219]	Default value for the attribute tcpGeneralInactivityTime of meta-class DoIpNetworkConfiguration
[TPS_MANI_03220]	Default value for the attribute vehicleAnnouncementCount of meta-class DoIpNetworkConfiguration
[TPS_MANI_03221]	Default value for the attribute vehicleAnnouncementInterval of meta-class DoIpNetworkConfiguration

▽

△

Number	Heading
[TPS_MANI_03222]	Default value for the attribute tcpAliveCheckResponseTimeout of meta-class DoIpNetworkConfiguration
[TPS_MANI_03553]	Applicability of health channel to a specific Process
[TPS_MANI_03554]	Several SomeipServiceInstanceToMachineMappings with equal settings
[TPS_MANI_03555]	Mix of SomeipServiceInstanceToMachineMapping and signal-based communication
[TPS_MANI_03556]	DDS-RPC Service Binding
[TPS_MANI_03557]	DDS ClientServerOperation Binding
[TPS_MANI_03558]	DDS Field Binding
[TPS_MANI_03559]	Definition of DdsProvidedServiceInstance.methodQosProps
[TPS_MANI_03560]	qosProfile of DdsProvidedServiceInstance.methodQosProps is optional
[TPS_MANI_03561]	Definition of DdsProvidedServiceInstance.fieldNotifierQosProps
[TPS_MANI_03562]	qosProfile of DdsProvidedServiceInstance.fieldNotifierQosProps is optional
[TPS_MANI_03563]	Definition of DdsProvidedServiceInstance.fieldGetSetQosProps
[TPS_MANI_03564]	qosProfile of DdsProvidedServiceInstance.fieldGetSetQosProps is optional
[TPS_MANI_03565]	Definition of DdsRequiredServiceInstance.methodQosProps
[TPS_MANI_03566]	qosProfile of DdsRequiredServiceInstance.methodQosProps is optional
[TPS_MANI_03567]	Definition of DdsRequiredServiceInstance.fieldNotifierQosProps
[TPS_MANI_03568]	qosProfile of DdsRequiredServiceInstance.fieldNotifierQosProps is optional
[TPS_MANI_03569]	Definition of DdsRequiredServiceInstance.fieldGetSetQosProps
[TPS_MANI_03570]	qosProfile of DdsRequiredServiceInstance.fieldGetSetQosProps is optional
[TPS_MANI_03571]	transportPlugin for DdsProvidedServiceInstance
[TPS_MANI_03572]	transportPlugin for DdsRequiredServiceInstance

Table E.15: Added Traceables in R18-10

E.4.2 Changed Traceables in R18-10

Number	Heading
[TPS_MANI_01001]	Meaning of ServiceInterface
[TPS_MANI_01041]	Startup configuration supports the definition of a launch sequence dependency

▽

△

Number	Heading
[TPS_MANI_01097]	Assignment of TLV data ids
[TPS_MANI_01100]	Semantics of `Allocator`
[TPS_MANI_01147]	Semantics of `PersistencyKeyValueDatabase.updateStrategy`
[TPS_MANI_01151]	Semantics of `PersistencyFileArray.updateStrategy`
[TPS_MANI_01166]	Semantics of `CppImplementationDataType`
[TPS_MANI_01176]	Standardized value for attribute `CppImplementationDataType.typeEmitter`
[TPS_MANI_01177]	Semantics of `CppImplementationDataType.typeEmitter`
[TPS_MANI_01180]	Collection of data types that requires serialization support
[TPS_MANI_03001]	Mapping of `AdaptivePlatformServiceInstance` to a `MachineDesign`
[TPS_MANI_03011]	Server Timing configuration for a `ProvidedSomeipServiceInstance`
[TPS_MANI_03021]	Requirements on the searched minor version from the client's point of view
[TPS_MANI_03025]	Client Timing configuration for a `RequiredSomeipServiceInstance`
[TPS_MANI_03070]	Size of a length field for a chosen array or map
[TPS_MANI_03103]	Default size for all array and map length fields
[TPS_MANI_03124]	`ServiceInterface.event` to `ISignalTriggering` mapping
[TPS_MANI_03125]	`ServiceInterface.method` to `ISignalTriggering`s mapping
[TPS_MANI_03126]	`ServiceInterface.field` mapping to `ISignalTriggering`s
[TPS_MANI_03134]	Configuration of supported TLS ciphersuites
[TPS_MANI_03137]	`ServiceInterfaceElementSecureComConfig` is not relevant in case of TLS communication
[TPS_MANI_03157]	Enabling of data collection for upd data transmission
[TPS_MANI_03158]	Configuration of a data collection on a `ProvidedServiceInstance` for transmission over udp
[TPS_MANI_03165]	Network Interface configuration for DoIP
[TPS_MANI_03170]	`CppImplementationDataType` of `category ARRAY`
[TPS_MANI_03173]	Definition of a multidimensional Array
[TPS_MANI_03175]	`CppImplementationDataType` of `category VECTOR`
[TPS_MANI_03177]	Definition of a multidimensional Vector
[TPS_MANI_03178]	`StdCppImplementationDataType` of `category STRING`
[TPS_MANI_03179]	C++ language binding of `StdCppImplementationDataType`s of `category STRING`
[TPS_MANI_03180]	Definition of Structures
[TPS_MANI_03181]	Definition of members in `StdCppImplementationDataType` of `category STRUCTURE`
[TPS_MANI_03184]	`CppImplementationDataType` of `category ASSOCIATIVE_MAP`
[TPS_MANI_03185]	Structure of an `CppImplementationDataType` of `category ASSOCIATIVE_MAP`
[TPS_MANI_03187]	Definition of enumeration types

▽

△

Number	Heading
[TPS_MANI_03193]	`CppImplementationDataType` of `category` `TYPE_REFERENCE`
[TPS_MANI_03196]	Semantics of `CppImplementationDataTypeElementQualifier.inplace` attribute
[TPS_MANI_03503]	Applicability of checkpoints to a specific `Process`
[TPS_MANI_03512]	Applicability of global supervision without `Process` context
[TPS_MANI_03516]	Condition evaluation for `HealthChannelSupervision`
[TPS_MANI_03518]	`PhmLogicalExpression` definition
[TPS_MANI_03519]	`PhmRule` definition
[TPS_MANI_03520]	Execution of `PhmActionList` with `actionListExecution=triggeredOnEvaluation`
[TPS_MANI_03521]	Execution of `PhmActionList` with `actionListExecution=triggeredOnChange`
[TPS_MANI_03522]	Definition of actions for application software
[TPS_MANI_03523]	Definition of actions for Platform Instance
[TPS_MANI_03524]	Definition of actions for Watchdog
[TPS_MANI_03526]	DDS `VariableDataPrototype` binding
[TPS_MANI_03527]	Definition of `DdsProvidedServiceInstance`
[TPS_MANI_03528]	Definition of `DdsProvidedServiceInstance.eventQosProps`
[TPS_MANI_03529]	Definition of `DdsRequiredServiceInstance`
[TPS_MANI_03530]	Definition of `DdsRequiredServiceInstance.eventQosProps`
[TPS_MANI_03531]	`qosProfile` of `DdsProvidedServiceInstance.eventQosProps` is optional
[TPS_MANI_03532]	`qosProfile` of `DdsRequiredServiceInstance.eventQosProps` is optional
[TPS_MANI_03533]	`DdsServiceInstanceToMachineMapping`
[TPS_MANI_03552]	Supervision cycle for `GlobalSupervision`

Table E.16: Changed Traceables in R18-10

E.4.3 Deleted Traceables in R18-10

Number	Heading
[TPS_MANI_01008]	Semantics of `ExecutableGroup`
[TPS_MANI_01009]	Standardized values of `ExecutableGroup.category`
[TPS_MANI_01018]	`ImplementationDataType` of `category` `VECTOR`
[TPS_MANI_01028]	`ImplementationDataType` of `category` `ASSOCIATIVE_MAP`
[TPS_MANI_01029]	Usage of `ImplementationDataType`
[TPS_MANI_01030]	`ImplementationDataType` of `category` `STRING`
[TPS_MANI_01042]	Definition of a linear `ImplementationDataType` of `category` `VECTOR`

▽

△

Number	Heading
[TPS_MANI_01043]	Definition of a rectangular ImplementationDataType of category VECTOR
[TPS_MANI_01044]	Structure of an ImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_01055]	Definition of application-level errors
[TPS_MANI_01056]	Semantics of ApplicationError.errorContext
[TPS_MANI_01058]	Ability to create a mapping of ApplicationErrors aggregated in the role possibleError
[TPS_MANI_01062]	ImplementationDataType to generate a C++ enum
[TPS_MANI_01063]	Sharing of ImplementationDataType with enumeration semantics
[TPS_MANI_01074]	Specification of encryption of persistent data
[TPS_MANI_01075]	Specification of redundancy of persistent data
[TPS_MANI_01077]	Specification of file encryption
[TPS_MANI_01082]	Eligibility of DataPrototypes for the definition of optionality
[TPS_MANI_01083]	Optionality is supported for ApplicationDataType as well as ImplementationDataType
[TPS_MANI_01084]	Optionality for a DataPrototype typed by an ApplicationDataType
[TPS_MANI_01085]	Definition of optionality for a DataPrototype typed by an ImplementationDataType
[TPS_MANI_01087]	Interaction with crypto software
[TPS_MANI_01088]	Semantics of CryptoNeed
[TPS_MANI_01089]	Relation between CryptoNeed and PortPrototype
[TPS_MANI_01090]	Modeling of crypto software as a platform module
[TPS_MANI_01091]	Semantics of CryptoJob
[TPS_MANI_01092]	Mapping between CryptoNeed and CryptoJob
[TPS_MANI_01093]	Semantics of CryptoDriver
[TPS_MANI_01094]	Scope of CryptoDriver
[TPS_MANI_01095]	Semantics of CryptoKeySlot
[TPS_MANI_01096]	Semantics of the CryptoPrimitive
[TPS_MANI_01098]	Constraints on the definition of an ImplementationDataType of category VECTOR
[TPS_MANI_01099]	Semantics of ImplementationDataTypeElementExtension
[TPS_MANI_01101]	Size-constrained allocation of memory
[TPS_MANI_01102]	Specification of a namespace for an ImplementationDataType of category VECTOR
[TPS_MANI_01133]	Optional element of an event
[TPS_MANI_01134]	Optional element in the context of a method
[TPS_MANI_03121]	Signal-based VariableDataPrototype binding
[TPS_MANI_03122]	Signal-based Field binding

▽

△

Number	Heading
[TPS_MANI_03123]	Signal-based ClientServerOperation binding
[TPS_MANI_03135]	Configuration of TLS PSK Identity
[TPS_MANI_03136]	Configuration of requirements for the TLS cryptographic job
[TPS_MANI_03141]	Mapping between SecOcJobRequirement and CryptoJob
[TPS_MANI_03142]	Mapping between TlsJobRequirement and CryptoJob
[TPS_MANI_03143]	Mapping between PresharedKeyIdentity and CryptoKeySlot
[TPS_MANI_03144]	C++ language binding of ImplementationDataTypes of category STRING
[TPS_MANI_03182]	Definition of members in CppImplementationDataTypeElement of category STRUCTURE

Table E.17: Deleted Traceables in R18-10

E.4.4 Added Constraints in R18-10

Number	Heading
[constr_1593]	Completeness of the existence of a set of TlvDataIdDefinition.tlvArguments
[constr_1594]	Consistent assignment of TLV data ids to ApplicationRecordDataType
[constr_1595]	Consistent assignment of TLV data ids to CppImplementationDataType or CppImplementationDataTypeElement
[constr_1596]	Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to ArgumentDataPrototype
[constr_1597]	Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to ApplicationRecordElement
[constr_1598]	Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to CppImplementationDataTypeElement
[constr_1599]	TlvDataIdDefinition referencing ArgumentDataPrototype
[constr_1600]	TlvDataIdDefinition referencing ApplicationRecordElement
[constr_1601]	TlvDataIdDefinition referencing CppImplementationDataTypeElement
[constr_1603]	Completeness of the existence of a set of TlvDataIdDefinition.tlvRecordElements
[constr_1604]	Completeness of the existence of a set of TlvDataIdDefinition.tlvSubElements
[constr_1605]	Standardized values of attribute Executable.category
[constr_1606]	Processes with mutual ExecutionDependencys
[constr_1613]	File name of matching pairs of PersistencyFileProxy and PersistencyFile
[constr_1614]	Existence of attribute TransformationPropsToServiceInterfaceElementMapping.transformationProps.sessionHandling
[constr_1615]	Existence of attribute SomeipDataPrototypeTransformationProps.someipTransformationProps.sessionHandling
[constr_1618]	Ability to shut down

▽

△

Number	Heading
[constr_1619]	Ability to restart
[constr_1620]	Value of schedulingPriority if schedulingPolicy is set to schedulingPolicyFifo or schedulingPolicyRoundRobin
[constr_1621]	Value of schedulingPriority if StartupConfig.schedulingPolicy is set to schedulingPolicyOther
[constr_1625]	Existence of reference ApApplicationError.errorDomain
[constr_1627]	Supported value range for attribute ApApplicationErrorDomain.value
[constr_1628]	Definition of static length field sizes in case of TLV usage
[constr_1629]	Identical sizes of length fields in case of TLV usage
[constr_1630]	No definition of length field sizes on DataPrototype level in case of TLV usage
[constr_1658]	Number of DiagnosticTroubleCodeUdsToClearConditionGroupMapping elements per DiagnosticTroubleCodeUds
[constr_1659]	Restriction for the usage of CppImplementationDataTypeElementQualifier.inplace
[constr_1660]	Restriction for the usage of CppTemplateArgument.inplace
[constr_1661]	Multiplicity of OsModuleInstantiation.resourceGroup
[constr_1663]	Standardized values of attribute DiagnosticServiceDataIdentifierPortMapping.category
[constr_1664]	Unique ApApplicationError.shortName
[constr_1665]	Unique ApApplicationError.errorCode
[constr_1666]	References from PersistencyPortPrototypeToKeyValueDatabaseMapping to PersistencyKeyValueDatabase
[constr_1667]	References from PersistencyPortPrototypeToFileArrayMapping to PersistencyFileArray
[constr_1668]	Allowed combinations of PersistencyRedundancyCrc.length and algorithmFamily
[constr_1673]	Existence of attributes hasGetter, hasSetter, and hasNotifier
[constr_1674]	Suported encoding of StdCppImplementationDataType of category STRING
[constr_1675]	Existence of attribute ApSomeipTransformationProps.stringEncoding
[constr_1676]	Consistency of references shallRunOn and shallNotRunOn
[constr_1677]	Mutual exclusive existence of references shallRunOn and shallNotRunOn
[constr_1678]	Allowed values for attribute ApSomeipTransformationProps.stringEncoding
[constr_3443]	Specification of a namespace for a StdCppImplementationDataType
[constr_3446]	CppTemplateArgument with allocator reference and the inplace flag
[constr_3447]	ApSomeipTransformationProps.sizeOfArrayLengthField that equals 0
[constr_3462]	CppTemplateArgument.templateType reference to StdCppImplementationDataType of category STRUCTURE and the inplace flag
[constr_3485]	UDP endpoint using DTLS can only serve provided or required service instances exclusively
[constr_3486]	TCP endpoint using TLS can only serve provided or required service instances exclusively.

▽

△

Number	Heading
[constr_3487]	TCP endpoint can only serve provided or required service instances exclusively
[constr_3492]	DoIpInstantiation.logicalAddress shall be defined as member in the DoIpRequestConfiguration
[constr_3493]	Applicable attributes for standardized E2E Profiles
[constr_3494]	Mandatory Machine States
[constr_3495]	Supported value range for attribute DoIpInstantiation.eid
[constr_3496]	Supported value range for attribute DoIpInstantiation.gid
[constr_3497]	Supported value range for attribute DoIpInstantiation.maxRequestBytes
[constr_3498]	Supported value range for attribute DoIpInstantiation.logicalAddress
[constr_3499]	Supported value range for attribute DoIpRequestConfiguration.startAddress
[constr_3537]	LocalSupervision referenced once in the context of a GlobalSupervision
[constr_3538]	Only one ServiceInstanceToMachineMapping per technology and CommunicationConnector
[constr_3539]	Only one AliveSupervision per SupervisionCheckpoint
[constr_3540]	SupervisionCheckpoint in supervision graph
[constr_3541]	qosProfile mandatory for DdsProvidedServiceInstance
[constr_3542]	qosProfile mandatory for DdsRequiredServiceInstance
[constr_3543]	At least one transportPlugin definition required for each DdsProvidedServiceInstance
[constr_3544]	At least one transportPlugin definition required for each DdsRequiredServiceInstance
[constr_5000]	Supported value range for attribute DoIpRequestConfiguration.endAddress
[constr_5001]	Usage of DoIpNetworkConfiguration.eidUseMac
[constr_5002]	Supported values of ServiceInstanceToMachineMapping.category
[constr_5003]	Existence of TlsCryptoCipherSuite.certificate in the *server* role
[constr_5004]	Mapping of a Process to a Machine is mandatory in the Execution Manifest

Table E.18: Added Constraints in R18-10

E.4.5 Changed Constraints in R18-10

Number	Heading
[constr_1490]	Allowed value of category for reference ProcessToMachineMapping.process.executable
[constr_1551]	Existence of DataPrototypeInServiceInterfaceRef.dataPrototype vs. DataPrototypeInServiceInterfaceRef.elementInImplDatatype
[constr_1572]	Usage of SwDataDefProps.implementationDataType within a CppImplementationDataType
[constr_1573]	CppTemplateArgument.isVariadicTemplate is set to True

▽

△

Number	Heading
[constr_1582]	PersistencyKeyValuePair.valueDataType shall match to AbstractImplementationDataType for the corresponding PersistencyDataElement
[constr_1585]	Standardized values of attribute DiagnosticServiceSwMapping.category
[constr_1589]	Value of file.fileName
[constr_3375]	method with attribute fireAndForget set to true shall not reference an ApApplicationError
[constr_3392]	ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are mandatory in case of SecOC communication
[constr_3433]	Aggregation of templateArguments for an ARRAY
[constr_3434]	Aggregation of templateArguments for a VECTOR
[constr_3527]	PhmLogicalExpression referenced by one PhmRule
[constr_3528]	Value range of domainId
[constr_3529]	Value range of serviceInstanceId

Table E.19: Changed Constraints in R18-10

E.4.6 Deleted Constraints in R18-10

Number	Heading
[constr_1474]	SwDataDefProps applicable to ImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*
[constr_1475]	ImplementationDataType of category STRING is limited
[constr_1476]	ImplementationDataType of category VECTOR is limited
[constr_1477]	ImplementationDataType of category ASSOCIATIVE_MAP is limited
[constr_1479]	No support for certain values of ImplementationDataType.category
[constr_1484]	Applicability of ModeDependentStartupConfig.executionDependency
[constr_1485]	No subElement for ImplementationDataType of category STRING
[constr_1486]	ImplementationDataType of category STRING and SwBaseType
[constr_1487]	Number of subElements of an ImplementationDataType of category ASSOCIATIVE_MAP
[constr_1491]	Semantics of ServiceInterface.possibleError
[constr_1493]	ArgumentDataPrototype referenced in the role ApplicationError.errorContext
[constr_1495]	Initial value for field
[constr_1506]	ImplementationDataType of category VECTOR shall not define dynamicArraySizeProfile
[constr_1508]	BaseTypeDirectDefinition.nativeDeclaration shall not be set to the value enum
[constr_1522]	Semantics of ClientServerOperation.possibleError

▽

△

Number	Heading
[constr_1527]	ImplementationDataTypeElement finally referenced as the target element in the context of an ImplementationDataTypeElementInAutosarDataPrototypeRef
[constr_1528]	Definition of optionality for multiple DataPrototypes typed by the same AutosarDataType
[constr_1529]	Standardized values of CryptoNeed.category
[constr_1530]	Standardized values of CryptoPrimitive.algorithmFamily and CryptoKeySlot.algorithmFamily
[constr_1531]	Standardized values of CryptoPrimitive.algorithmMode
[constr_1532]	Consistent assignment of TLV data ids to data structures with optional members
[constr_1537]	Consistent assignment of TLV data ids to arguments of a given ClientServerOperation
[constr_1546]	Existence of attributes of ServiceInterfaceSubElement
[constr_1547]	Reference from ImplementationDataTypeExtension to ImplementationDataType
[constr_1548]	Reference from ImplementationDataTypeElementExtension to ImplementationDataTypeElement
[constr_1577]	Specification of a nativeDeclaration for a CppImplementationDataType
[constr_1587]	DiagnosticServiceSwMapping.category set to DATA_IDENTIFIER
[constr_1588]	DiagnosticServiceSwMapping.category set to GENERIC_UDS_SERVICE
[constr_3293]	Mandatory information of a RequiredSomeipServiceInstance
[constr_3303]	ANY not allowed for SomeipServiceInterfaceDeployment.serviceInterfaceVersion
[constr_3350]	Consistent value of category for ExecutableGroups referencing an Executable
[constr_3377]	Restriction of ISignalTriggering references in SignalBasedFieldToISignalTriggeringMapping
[constr_3422]	CppImplementationDataType of category STRING and SwBaseType
[constr_3428]	Structure shall own at least one element
[constr_3432]	Allowed subElements for Structures

Table E.20: Deleted Constraints in R18-10

E.5 Constraint and Specification Item History of this document according to AUTOSAR Release R19-03

E.5.1 Added Traceables in R19-03

Number	Heading
[TPS_MANI_01211]	Specification of executable software within SoftwareClusterDesign
[TPS_MANI_01212]	Usage of attribute typeEmitter in the context of a CustomCppImplementationDataType
[TPS_MANI_01213]	Semantics of meta-class StrongRevisionLabelString
[TPS_MANI_01214]	Semantics of SoftwareCluster.conflictsTo
[TPS_MANI_01215]	Semantics of meta-class SoftwareActivationDependencyFormula
[TPS_MANI_01216]	Semantics of meta-class SoftwareActivationDependencyFormulaPart
[TPS_MANI_01217]	Semantics of metaclass SoftwareActivationDependencyCompareCondition
[TPS_MANI_01218]	Cryptographic signature of SoftwareCluster
[TPS_MANI_01219]	License of software in included SoftwareCluster
[TPS_MANI_01220]	Release notes of software in included SoftwareCluster
[TPS_MANI_01221]	Semantics of meta-class SoftwarePackage
[TPS_MANI_01222]	Cryptographic signature of SoftwarePackage
[TPS_MANI_01223]	Semantics of attribute SoftwarePackage.packagerId
[TPS_MANI_01224]	Actions taken after installation of a SoftwarePackage
[TPS_MANI_01225]	Actions taken during installation of a SoftwarePackage
[TPS_MANI_01226]	Machine-specific configuration settings for the UCM module
[TPS_MANI_01227]	Semantics of attribute UcmModuleInstantiation.identifier
[TPS_MANI_01228]	Semantics of meta-class ProcessDesign
[TPS_MANI_01229]	Pre-allocation of a given ProcessDesign on a specific MachineDesign
[TPS_MANI_01230]	Semantics of DiagnosticProvidedDataMapping
[TPS_MANI_01231]	GrantDesign references ProcessDesign
[TPS_MANI_01232]	Semantics of meta-class ComOfferServiceGrantDesign
[TPS_MANI_01233]	Semantics of meta-class ComFindServiceGrantDesign
[TPS_MANI_01234]	Semantics of ComFieldGrantDesign
[TPS_MANI_01235]	Semantics of ComEventGrantDesign
[TPS_MANI_01236]	Semantics of ComMethodGrantDesign
[TPS_MANI_01237]	Semantics of meta-class ComFieldGrant
[TPS_MANI_01238]	Semantics of meta-class ComMethodGrant
[TPS_MANI_01239]	Semantics of meta-class ComEventGrant
[TPS_MANI_01240]	Semantics of meta-class ComOfferServiceGrant
[TPS_MANI_01241]	Semantics of meta-class ComFindServiceGrant
[TPS_MANI_01242]	PortInterfaces used for communication with the AUTOSAR Diagnostic Manager

△

Number	Heading
[TPS_MANI_01243]	Semantics of DiagnosticDataIdentifierInterface
[TPS_MANI_01244]	Semantics of DiagnosticDataElementInterface
[TPS_MANI_01245]	Semantics of DiagnosticDataIdentifierGenericInterface
[TPS_MANI_01246]	Semantics of DiagnosticMonitorInterface
[TPS_MANI_01247]	Semantics of DiagnosticDTCInformationInterface
[TPS_MANI_01248]	Semantics of DiagnosticEventInterface
[TPS_MANI_01249]	Semantics of DiagnosticConditionInterface
[TPS_MANI_01250]	Semantics of DiagnosticIndicatorInterface
[TPS_MANI_01251]	Semantics of DiagnosticSecurityLevelInterface
[TPS_MANI_01252]	Semantics of DiagnosticServiceValidationInterface
[TPS_MANI_01253]	Semantics of DiagnosticOperationCycleInterface
[TPS_MANI_01254]	Semantics of DiagnosticGenericUdsInterface
[TPS_MANI_01255]	Semantics of DiagnosticGenericUdsInterface
[TPS_MANI_01256]	AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticIndicatorInterface
[TPS_MANI_01257]	AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticConditionInterface
[TPS_MANI_01258]	AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticGenericUdsInterface
[TPS_MANI_01259]	Mapping of DiagnosticClearCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01260]	Mapping of DiagnosticIndicator to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01261]	Mapping of DiagnosticMemoryDestination to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01262]	Mapping of DiagnosticSecurityLevel to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01263]	Mapping of DiagnosticDataIdentifier to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01264]	Mapping of DiagnosticServiceInstance to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01265]	Semantics of DiagnosticDownloadInterface and DiagnosticDownloadInterface
[TPS_MANI_01266]	Mapping of DiagnosticServiceInstance for upload/download to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_03223]	Existence of CppImplementationDataType
[TPS_MANI_03224]	Modeling of a Partial Network Cluster
[TPS_MANI_03225]	References to vlans in PncMapping
[TPS_MANI_03226]	Collection of partialNetworks and vlans in NmNetworkHandle

Table E.21: Added Traceables in R19-03

E.5.2 Changed Traceables in R19-03

Number	Heading
[TPS_MANI_01012]	Formal modeling of application startup behavior
[TPS_MANI_01013]	Semantics of meta-class StateDependentStartupConfig
[TPS_MANI_01017]	Relation of startup configuration to resource group
[TPS_MANI_01041]	Startup configuration supports the definition of a launch sequence dependency
[TPS_MANI_01046]	Semantics of StateDependentStartupConfig.functionGroupState
[TPS_MANI_01049]	Mapping of DiagnosticOperationCycle to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01050]	Mapping of DiagnosticEnableCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01051]	Mapping of DiagnosticStorageCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01136]	AutosarDataPrototype is the target of the DataPrototypeInServiceInterfaceRef
[TPS_MANI_01137]	Applicable use cases for DataPrototypeInServiceInterfaceRef
[TPS_MANI_01164]	Semantics of SoftwareCluster.dependsOn
[TPS_MANI_01177]	Semantics of CppImplementationDataType.typeEmitter
[TPS_MANI_01207]	Standardized values of attribute PersistencyRedundancyCrc.algorithmFamily
[TPS_MANI_03070]	Size of a length field for a chosen array or map
[TPS_MANI_03071]	Size of a length field for a chosen structure
[TPS_MANI_03072]	Size of a length field for a chosen union
[TPS_MANI_03073]	Alignment of a dynamic DataPrototype
[TPS_MANI_03074]	Size of a type selector field for a chosen union
[TPS_MANI_03075]	Byte Order of chosen DataPrototype in the serialized data stream
[TPS_MANI_03116]	Size of a length field for a chosen string
[TPS_MANI_03127]	Usage of End2EndEventProtectionProps
[TPS_MANI_03128]	Usage of same dataId in case of Multi-Binding
[TPS_MANI_03152]	Assignment of a StateDependentStartupConfig to a function group state
[TPS_MANI_03187]	Definition of enumeration types
[TPS_MANI_03190]	CppImplementationDataType of category VARIANT
[TPS_MANI_03202]	Definition of bitfield types
[TPS_MANI_03217]	On-the-wire encoding for a chosen string

Table E.22: Changed Traceables in R19-03

E.5.3 Deleted Traceables in R19-03

Number	Heading
[TPS_MANI_01038]	Diagnostic software mapping on the *AUTOSAR adaptive platform*
[TPS_MANI_01170]	Semantics of CppTemplateArgument.isVariadicTemplate
[TPS_MANI_01181]	Use cases for the application of DiagnosticServiceSwMapping
[TPS_MANI_01193]	Combination of ModeDependentStartupConfig.machineMode and ModeDependentStartupConfig.functionGroupMode
[TPS_MANI_03066]	Description of machine states
[TPS_MANI_03153]	Semantics of ModeDependentStartupConfig.functionGroupMode

Table E.23: Deleted Traceables in R19-03

E.5.4 Added Constraints in R19-03

Number	Heading
[constr_1687]	Definition of machine state
[constr_1688]	StateDependentStartupConfig shall only refer to function group states of the same function group
[constr_1689]	Modeling of a startup dependency between different Processes
[constr_1690]	SoftwareCluster shall only be referenced by a single SoftwarePackage.
[constr_1691]	UcmModuleInstantiation.identifier shall be unique
[constr_1692]	Value of schedulingPriority
[constr_1693]	Relation of Executable, ProcessDesign, and Process
[constr_1695]	Semantics of a Grant depends on the existence of IamModuleInstantiation
[constr_1696]	ClientServerOperation aggregated by DiagnosticRoutineInterface
[constr_1697]	Restriction for ClientServerOperation aggregated by a DiagnosticDataIdentifierInterface or DiagnosticDataElementInterface
[constr_1698]	Target SwcServiceDependency of DiagnosticClearConditionPortMapping.swcServiceDependencyInExecutable
[constr_1699]	Target SwcServiceDependency of DiagnosticIndicatorPortMapping.swcServiceDependencyInExecutable
[constr_1700]	Target SwcServiceDependency of DiagnosticMemoryDestinationPortMapping.swcServiceDependencyInExecutable
[constr_1701]	Target SwcServiceDependency of DiagnosticSecurityLevelPortMapping.swcServiceDependencyInExecutable
[constr_1702]	Target SwcServiceDependency of DiagnosticServiceDataIdentifierPortMapping.swcServiceDependencyInExecutable
[constr_1703]	Target SwcServiceDependency of DiagnosticGenericUdsPortMapping.swcServiceDependencyInExecutable
[constr_1704]	Target SwcServiceDependency of DiagnosticUploadDownloadPortMapping.swcServiceDependencyInExecutable
[constr_5033]	Compatibility of data types with category VALUE
[constr_5034]	Compatibility of data types with category BOOLEAN

▽

△

Number	Heading
[constr_5035]	Compatibility of data types with `category STRING`
[constr_5036]	Compatibility of data types with `category ARRAY`
[constr_5037]	Compatibility of data types with `category ARRAY` with `variableSize`
[constr_5038]	Compatibility of data types with `category ARRAY` with `fixedSize`
[constr_5039]	Compatibility of data types with `category STRUCTURE`
[constr_5040]	Compatibility of `ApplicationRecordDataType` and `CppImplementationDataType` that both represent an `Optional Element Structure`
[constr_5041]	Compatibility of data types with `category ASSOCIATIVE_MAP`
[constr_5042]	No data type mapping for `CppImplementationDataType` of `category VARIANT`
[constr_5043]	Forbidden mappings to `CppImplementationDataType`
[constr_5044]	`DataTypeMap` for composite data types
[constr_5045]	Only one `SomeipServiceDiscovery` configuration per VLAN is allowed
[constr_5046]	Usage of `DoIpNetworkConfiguration.eidUseMac`
[constr_5047]	Supported values of `ServiceInstanceToMachineMapping.category`
[constr_5048]	Existence of `TlsCryptoCipherSuite.certificate` in the *server* role

Table E.24: Added Constraints in R19-03

E.5.5 Changed Constraints in R19-03

Number	Heading
[constr_1481]	Usage of `DataPrototypeInServiceInterfaceRef` in the *AUTOSAR adaptive platform*
[constr_1500]	Target `SwcServiceDependency` of `DiagnosticEventPortMapping.swcServiceDependencyInExecutable`
[constr_1501]	Target `SwcServiceDependency` of `DiagnosticOperationCyclePortMapping.swcServiceDependencyInExecutable`
[constr_1502]	Target `SwcServiceDependency` of `DiagnosticEnableConditionPortMapping.swcServiceDependencyInExecutable`
[constr_1503]	Target `SwcServiceDependency` of `DiagnosticStorageConditionPortMapping.swcServiceDependencyInExecutable`
[constr_1551]	Existence of `DataPrototypeInServiceInterfaceRef.dataPrototype` vs. `DataPrototypeInServiceInterfaceRef.elementInImplDatatype`
[constr_1567]	Existence of `SoftwareCluster.subSoftwareCluster` and `SoftwareCluster.dependsOn/conflictsTo`
[constr_1595]	Consistent assignment of TLV data ids to `CppImplementationDataType` or `CppImplementationDataTypeElement`
[constr_1606]	`Process`es with mutual `ExecutionDependency`s
[constr_1615]	Existence of attribute `SomeipDataPrototypeTransformationProps.someipTransformationProps.sessionHandling`
[constr_1618]	Ability to shut down

▽

△

Number	Heading
[constr_1619]	Ability to restart
[constr_3396]	Number of Process.stateDependentStartupConfig that refer to the same functionGroupState
[constr_3413]	StateDependentStartupConfig of a Process is mapped to exactly one ResourceGroup
[constr_3421]	Fibex elements applicable for a System of category MACHINE_DESIGN_EXTRACT
[constr_3423]	StateDependentStartupConfig of a Process shall reference a functionGroupState
[constr_3424]	StateDependentStartupConfig shall never reference the functionGroupState Off
[constr_3447]	ApSomeipTransformationProps.sizeOfArrayLengthField that equals 0

Table E.25: Changed Constraints in R19-03

E.5.6 Deleted Constraints in R19-03

Number	Heading
[constr_1499]	Target SwcServiceDependency of DiagnosticServiceSwMapping.mappedSwcServiceDependencyInExecutable
[constr_1504]	Number of Process.modeDependentStartupConfig that refer to the same machineMode
[constr_1573]	CppTemplateArgument.isVariadicTemplate is set to True
[constr_1574]	Number of CppTemplateArguments with isVariadicTemplate set to True
[constr_1575]	Position of CppTemplateArgument with isVariadicTemplate set to True
[constr_1585]	Standardized values of attribute DiagnosticServiceSwMapping.category
[constr_1586]	DiagnosticServiceSwMapping.category set to DATA_ELEMENT
[constr_1620]	Value of schedulingPriority if schedulingPolicy is set to schedulingPolicyFifo or schedulingPolicyRoundRobin
[constr_1621]	Value of schedulingPriority if StartupConfig.schedulingPolicy is set to schedulingPolicyOther
[constr_1663]	Standardized values of attribute DiagnosticServiceDataIdentifierMapping.category
[constr_3380]	End2EndEventProtectionProps shall not reference an event and a notifier at the same time
[constr_3397]	ModeDependentStartupConfig that refers to a functionGroupMode and to a machineMode
[constr_3398]	ModeDependentStartupConfig that refers to function group modes of different function groups
[constr_3494]	Mandatory Machine States
[constr_3531]	Mandatory definition of healthChannelId
[constr_3536]	Mandatory definition of supervisedEntityId
[constr_5001]	Usage of DoIpNetworkConfiguration.eidUseMac

▽

△

Number	Heading
[constr_5002]	Supported values of ServiceInstanceToMachineMapping.category
[constr_5003]	Existence of TlsCryptoCipherSuite.certificate in the *server* role

Table E.26: Deleted Constraints in R19-03

E.6 Constraint and Specification Item History of this document according to AUTOSAR Release R19-11

E.6.1 Added Traceables in R19-11

Number	Heading
[TPS_MAIN_01281]	Usage of meta-class RecoveryViaApplicationAction
[TPS_MANI_01267]	Semantics of attribute SoftwareClusterDesign.dependsOn
[TPS_MANI_01268]	Semantics of attribute SoftwareClusterDesign.conflictsTo
[TPS_MANI_01269]	Specification of boundaries for resource consumption
[TPS_MANI_01270]	Reference from TransformationPropsToServiceInterfaceElementMapping to TlvDataIdDefinitionSet
[TPS_MANI_01271]	Semantics of Executable.loggingBehavior
[TPS_MANI_01272]	Duplicate entries in logTraceLogMode
[TPS_MANI_01273]	Support for trusted Platform
[TPS_MANI_01274]	System category for a design description that has one single Adaptive Machine in scope
[TPS_MANI_01275]	Semantics of meta-class ServiceInstanceToSwClusterDesignPortPrototypeMapping
[TPS_MANI_01276]	Semantics of CompositionRPortToExecutableRPortMapping and CompositionPPortToExecutablePPortMapping
[TPS_MANI_01277]	Definition of a start-up timeout for a StartupConfig of a Process
[TPS_MANI_01278]	Definition of a termination timeout for a StartupConfig of a Process
[TPS_MANI_01279]	Semantics of Executable.reportingBehavior
[TPS_MANI_01280]	Semantics of meta-class PhmRecoveryActionInterface
[TPS_MANI_01282]	Semantics of reference CompositionPortToExecutablePortMapping.processDesign
[TPS_MANI_01283]	Semantics of meta-class RawDataStreamInterface
[TPS_MANI_01284]	Granularity of meta-class RawDataStreamGrantDesign
[TPS_MANI_01285]	Purpose of meta-class RawDataStreamDeployment
[TPS_MANI_01286]	Semantics of attribute RawDataStreamMethodDeployment.callTimeout
[TPS_MANI_01287]	Semantics of RawDataStreamMapping
[TPS_MANI_01288]	Impact of the SoftwarePackage on the value of function group states on the target platform

▽

△

Number	Heading
[TPS_MANI_01289]	Order of function group states is relevant
[TPS_MANI_01290]	`VehiclePackage` names affected UCMs
[TPS_MANI_01291]	Identification of an actual UCM in the context of an update campaign
[TPS_MANI_01292]	Definition of fallback-order for UCM master
[TPS_MANI_01294]	Update campaign depends on driver's acceptance
[TPS_MANI_01295]	Semantics of `VehicleRolloutStep`
[TPS_MANI_01296]	Ordered execution of rollout steps in an update campaign
[TPS_MANI_01297]	Semantics of meta-class `UcmStep`
[TPS_MANI_01298]	No ordering of `VehicleRolloutStep.ucmProcessing`
[TPS_MANI_01299]	Aggregation of `SoftwarePackageStep`s at `UcmStep`
[TPS_MANI_01300]	Semantics of reference `SoftwarePackageStep.transfer.transfer`
[TPS_MANI_01301]	Semantics of aggregation `SoftwarePackageStep.transfer`
[TPS_MANI_01302]	Semantics of reference `SoftwarePackageStep.process`
[TPS_MANI_01303]	Semantics of reference `SoftwarePackageStep.preActivate`
[TPS_MANI_01304]	Semantics of reference `SoftwarePackageStep.verify`
[TPS_MANI_01305]	Semantics of attribute `SoftwarePackageStep.activationSwitch`
[TPS_MANI_01306]	Simultaneous existence of attributes `SoftwarePackageStep.transfer` and `SoftwarePackageStep.process`
[TPS_MANI_01307]	Semantics of meta-class `EthernetRawDataStreamGrant`
[TPS_MANI_03227]	Usage of ephemeral ports
[TPS_MANI_03228]	Usage of `End2EndMethodProtectionProps`
[TPS_MANI_03229]	Usage of same `End2EndMethodProtectionProps.dataId` in case of Multi-Binding
[TPS_MANI_03230]	Sharing timers for `ProvidedSomeipServiceInstance`
[TPS_MANI_03231]	Sharing timers for `RequiredSomeipServiceInstance`
[TPS_MANI_03232]	Definition of general IPsec configuration settings
[TPS_MANI_03233]	IPsec mode
[TPS_MANI_03234]	IPsec AH and ESP CipherSuites
[TPS_MANI_03573]	Definition of no minimum deadline supervision
[TPS_MANI_03574]	Definition of no maximum deadline supervision
[TPS_MANI_03575]	Definition of no minimum alive supervision
[TPS_MANI_03576]	Definition of no maximum alive supervision
[TPS_MANI_03577]	`headerId` required for signal-service-translation
[TPS_MANI_03578]	Signal-based `ServiceInterface` binding over `Ethernet`
[TPS_MANI_03579]	Signal-based `ServiceEventDeployment` over `Ethernet`
[TPS_MANI_03580]	Service offer at startup
[TPS_MANI_03581]	Service find at startup
[TPS_MANI_03582]	Service find for required signal

▽

△

Number	Heading
[TPS_MANI_03583]	Service subscribe for required signal
[TPS_MANI_03584]	Definition of transmission triggers for translations with different sources
[TPS_MANI_03585]	Processing order of COM-Stack features
[TPS_MANI_03586]	No transmission trigger for translations with different sources
[TPS_MANI_03587]	Transmission trigger for translations with different sources
[TPS_MANI_03588]	Full translation before transmission triggering
[TPS_MANI_03589]	Reception data filter of COM-Stack
[TPS_MANI_03590]	Transfer properties and transmission modes of COM-Stack
[TPS_MANI_03591]	`SomeipEventDeployment.serializer` equals `signalBased`
[TPS_MANI_03592]	`ISignal` invalidation of COM-Stack
[TPS_MANI_03593]	`handleInvalid` = `dontInvalidate` behavior of COM-Stack
[TPS_MANI_03594]	`handleInvalid` = `replace` behavior of COM-Stack
[TPS_MANI_03595]	Update Bit support for `ISignal`
[TPS_MANI_03596]	Update Bit support for `ISignalIPdu`
[TPS_MANI_03597]	Support for `MultiplexedIPdu`
[TPS_MANI_03598]	Expected check period of E2E-Protected payload
[TPS_MANI_03599]	Expected update period of E2E-Protected payload
[TPS_MANI_03600]	Signal-service-translation of E2E protected payload
[TPS_MANI_03601]	Signal-service-translation of E2E protected payload - timeout handling
[TPS_MANI_03602]	Signal-service-translation of E2E protected payload - error handling
[TPS_MANI_03603]	Service-signal-translation of E2E protected payload
[TPS_MANI_03604]	Service-signal-translation of E2E protected payload - timeout handling
[TPS_MANI_03605]	Service-signal-translation of E2E protected payload - error handling
[TPS_MANI_03606]	Service offer for provided signal
[TPS_MANI_03607]	Handling of safe signal-service-translation in one `Executable`
[TPS_MANI_03608]	Support for safe signal-service-translation and service-signal-translation
[TPS_MANI_03609]	Support for safe signal-service-translation with same or different E2E profiles
[TPS_MANI_03610]	1:n mapping for E2E protected data
[TPS_MANI_03611]	E2E protected target out of E2E protected sources
[TPS_MANI_03612]	Sufficient ASIL level of translation software
[TPS_MANI_03614]	No translation of not OK E2E protected data out of several sources
[TPS_MANI_03615]	`SomeipEventDeployment.serializer` equals `someip`
[TPS_MANI_03616]	Semantic versioning of `ServiceInterface.majorVersion` and `ServiceInterface.minorVersion`
[TPS_MANI_03617]	Version mapping between `ServiceInterface` and `ServiceInterfaceDeployment`
[TPS_MANI_03618]	Usage of `RequiredSomeipServiceInstance.blacklistedVersion`
[TPS_MANI_03619]	SOME/IP Service search for `requiredMinorVersion`

▽

△

Number	Heading
[TPS_MANI_03620]	Service discovery control
[TPS_MANI_03621]	Data filter inside the signal-service-translation

Table E.27: Added Traceables in R19-11

E.6.2 Changed Traceables in R19-11

Number	Heading
[TPS_MANI_01032]	Usage of `ServiceInterfaceMapping`
[TPS_MANI_01057]	Semantics of `RPortPrototypeProps.searchIntention`
[TPS_MANI_01164]	Semantics of `SoftwareCluster.dependsOn`
[TPS_MANI_01196]	Semantics of `PersistencyDeployment.minimumSustainedSize`
[TPS_MANI_01197]	Semantics of `PersistencyDeployment.maximumAllowedSize`
[TPS_MANI_01214]	Semantics of `SoftwareCluster.conflictsTo`
[TPS_MANI_01215]	Semantics of meta-class `SoftwareActivationDependencyFormula`
[TPS_MANI_01216]	Semantics of meta-class `SoftwareActivationDependencyFormula-Part`
[TPS_MANI_01217]	Semantics of metaclass `SoftwareActivationDependencyCompare-Condition`
[TPS_MANI_01249]	Semantics of `DiagnosticConditionInterface`
[TPS_MANI_01255]	Semantics of DoIP `DiagnosticPortInterface`s
[TPS_MANI_03004]	IPv4 Multicast event destination address
[TPS_MANI_03010]	Udp Transport Protocol Configuration in case of IP-Multicast
[TPS_MANI_03061]	IPv6 Multicast event destination address
[TPS_MANI_03114]	Usage of `AssemblySwConnector`s in the System Design model
[TPS_MANI_03130]	Standardized `E2EProfileConfiguration.profileName` values
[TPS_MANI_03160]	Log and Trace configuration options in the Execution Manifest
[TPS_MANI_03161]	Log and Trace configuration options in the Service Instance Manifest
[TPS_MANI_03167]	Network configuration for Nm
[TPS_MANI_03205]	IPsec policy
[TPS_MANI_03206]	IPsec AH and ESP protocol configuration
[TPS_MANI_03207]	IPsec Internet Key Exchange protocol configuration
[TPS_MANI_03216]	Existence of `TlsCryptoCipherSuite.certificate` and `TlsCryptoCipherSuite.pskIdentity` in the *client* role

Table E.28: Changed Traceables in R19-11

E.6.3 Deleted Traceables in R19-11

Number	Heading
[TPS_MANI_01051]	Mapping of DiagnosticStorageCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01052]	Semantics of RPortPrototypeProps.portInstantiationBehavior
[TPS_MANI_01162]	Semantics of SoftwareClusterDesign.dependsOn
[TPS_MANI_03120]	Signal-based ServiceInterface binding
[TPS_MANI_03146]	Configuration of timeouts for a selected machine state or function group state
[TPS_MANI_03149]	Definition of a start-up timeout for a Process
[TPS_MANI_03150]	Definition of a termination timeout for a Process
[TPS_MANI_03550]	Usage of RPortPrototype for the interaction with Platform Health Management

Table E.29: Deleted Traceables in R19-11

E.6.4 Added Constraints in R19-11

Number	Heading
[constr_1705]	Target of reference SoftwareActivationDependencyCompareCondition.softwareActivationDependency
[constr_1707]	Eligible subclasses of HeapUsage in the context of StateDependentStartupConfig.resourceConsumption
[constr_1708]	Combination of CppImplementationDataTypeElement.isOptional and CppImplementationDataTypeElementQualifier.inplace
[constr_1709]	Applicability of attribute PersistencyRedundancyEnum.redundantPerKey
[constr_1710]	Consistency of values of attributes PersistencyInterface.redundancy and PersistencyRedundancyHandling.scope
[constr_1723]	ProvidedSomeipServiceInstance shall be unique in respect of serviceInstanceId, serviceInterfaceId and majorVersion
[constr_1727]	Qualified combinations of PortPrototypes and PhmSupervisedEntityInterface on application software level
[constr_1728]	Qualified combinations of PortPrototypes and PhmHealthChannelInterface on application software level
[constr_1729]	Qualified combinations of PortPrototypes and PhmRecoveryActionInterface on application software level
[constr_1730]	Restriction regarding the modeling of the PhmRecoveryActionInterface.recovery
[constr_1731]	Value of UcmDescription.identifier in the scope of a VehiclePackage
[constr_1732]	Existence of attribute activationSwitch set to True in the context of the enclosing UcmStep
[constr_1733]	Simultaneous existence of SoftwarePackageStep.preActivate and SoftwarePackageStep.verify
[constr_1734]	Restriction for attribute SoftwarePackageStep.activationSwitch

▽

△

Number	Heading
[constr_1736]	Multiplicity of reference LogicalSupervision.initialCheckpoint
[constr_1737]	Multiplicity of reference LogicalSupervision.finalCheckpoint
[constr_1738]	Multiplicity of reference GlobalSupervision.localSupervision
[constr_1739]	Multiplicity of aggregation LocalSupervision.transition
[constr_1740]	Multiplicity of reference LogicalSupervision.transition
[constr_1742]	Multiplicity of reference SupervisionCheckpoint.phmCheckpoint
[constr_3550]	Existence of ServiceInstanceToSignalMapping for an event with signal-Based serialization
[constr_3551]	Full mapping of target ISignalGroup
[constr_3552]	Full mapping of target event
[constr_3553]	Existence of ServiceInstanceToSignalMapping for an field with signal-Based serialization
[constr_3554]	E2E protection configuration check
[constr_3555]	No support for useAsCryptographicIPdu is true
[constr_3556]	Unique transport layer mapping
[constr_3557]	Mandatory majorVersion at SomeipServiceInterfaceDeployment.serviceInterfaceVersion
[constr_3558]	RequiredSomeipServiceInstance.blacklistedVersion is restricted to the usage of minorVersion
[constr_3561]	minimumMinorVersion and RequiredSomeipServiceInstance.requiredMinorVersion value
[constr_3562]	Existence of NonqueuedReceiverComSpec.filter
[constr_5052]	SOME/IP ServiceInstances of the same serviceInterface on one Machine
[constr_5056]	Restriction of CompositionSwComponentType.connector usage in AP
[constr_5057]	PassThroughSwConnector and ServiceInterfaceMapping
[constr_5102]	Usage of remote port ranges in IPSecRule is not allowed
[constr_5103]	Usage of local port ranges in IPSecRule is not allowed

Table E.30: Added Constraints in R19-11

E.6.5 Changed Constraints in R19-11

Number	Heading
[constr_1561]	Existence of SoftwareClusterDesign.subSoftwareCluster and SoftwareClusterDesign.dependsOn.dependentSoftwareClusterDesign
[constr_1567]	Existence of SoftwareCluster.subSoftwareCluster and SoftwareCluster.dependsOn/conflictsTo
[constr_1570]	Restriction for UserDefinedServiceInterfaceDeployment of category SERVICE_INTERFACE_DEPLOYMENT_IPC
[constr_1579]	SwDataDefProps applicable to CppImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*

▽

△

Number	Heading
[constr_1630]	No definition of length field sizes on DataPrototype level in case of TLV usage
[constr_3375]	method with attribute fireAndForget set to true shall not reference an ApApplicationError
[constr_3419]	Allowed usage of UdpNmNetworkConfiguration attributes
[constr_3426]	The logTraceFilePath is mandatory in case that logTraceLogMode is set to file
[constr_3427]	The logTraceFilePath is only relevant if logTraceLogMode is set to file
[constr_3493]	Applicable attributes for standardized E2E Profiles
[constr_5048]	Existence of TlsCryptoCipherSuite.certificate and TlsCryptoCipherSuite.pskIdentity in the *server* role

Table E.31: Changed Constraints in R19-11

E.6.6 Deleted Constraints in R19-11

Number	Heading
[constr_1503]	Target SwcServiceDependency of DiagnosticStorageConditionPortMapping.swcServiceDependencyInExecutable
[constr_3387]	Compatibility of PortPrototypes of different ServiceInterfaces
[constr_3388]	Compatibility of events
[constr_3389]	Compatibility of methods
[constr_3390]	Compatibility of fields
[constr_3411]	eventMulticastUdpPort, ipv4MulticastIpAddress and ipv6MulticastIpAddress not relevant for RequiredSomeipServiceInstances
[constr_3420]	System category for a design description that has one single Adaptive Machine in scope

Table E.32: Deleted Constraints in R19-11

E.7 Constraint and Specification Item History of this document according to AUTOSAR Release R20-11

E.7.1 Added Traceables in R20-11

Number	Heading
[TPS_MANI_01308]	Process is not designed for re-usability
[TPS_MANI_01309]	Semantics of attribute CppImplementationDataType.headerFile
[TPS_MANI_01310]	Semantics of SoftwareClusterDesign.dependsOn
[TPS_MANI_01311]	Handling of manufacturer checks
[TPS_MANI_01312]	Handling of supplier checks

▽

△

Number	Heading
[TPS_MANI_01313]	Definition of updateStrategy on element level
[TPS_MANI_01314]	Further qualification of properties of PortPrototypes typed by PersistencyKeyValueStorageInterfaces
[TPS_MANI_01315]	PersistencyKeyValuePair.initValue overrides PersistencyDataRequiredComSpec.initValue
[TPS_MANI_01316]	Existence of ServiceInstanceToPortPrototypeMapping.processDesign
[TPS_MANI_01317]	Existence of ServiceInstanceToPortPrototypeMapping.process
[TPS_MANI_01319]	Modeling of redundancy in the context of PersistencyInterface
[TPS_MANI_01320]	Definition of redundancy on interface level may be overruled in deployment
[TPS_MANI_01321]	Semantics of meta-class PersistencyDeploymentElement
[TPS_MANI_01322]	Semantics of meta-class PersistencyPortPrototypeToDeploymentMapping
[TPS_MANI_01323]	Matching pairs of PersistencyDataElement and PersistencyKeyValuePair
[TPS_MANI_01324]	Semantics of E2E attributes in ClientComSpec
[TPS_MANI_01325]	Semantics of E2E attributes in ServerComSpec
[TPS_MANI_01326]	Generic Mapping to a DiagnosticServiceInstance on the AUTOSAR Adaptive Platform
[TPS_MANI_01327]	Value of EndToEndTransformationComSpecProps.disableEndToEndCheck vs. value of EndToEndTransformationComSpecProps.disableEndToEndStateMachine
[TPS_MANI_01328]	Standardized values for attribute StartupConfig.schedulingPolicy
[TPS_MANI_01329]	Reference to model elements in different SoftwareClusters
[TPS_MANI_01330]	Definition of machine function group
[TPS_MANI_01331]	Standardized values of attribute SoftwareCluster.category
[TPS_MANI_01332]	Semantics of DiagnosticEcuResetInterface
[TPS_MANI_01333]	Attribute NotAvailableValueSpecification.defaultPattern is not applicable
[TPS_MANI_01334]	Semantics of StartupConfig.terminationBehavior
[TPS_MANI_01335]	Semantics of SoftwareClusterDependencyFormula.category
[TPS_MANI_01336]	Two use cases for using the DiagnosticEventPortMapping
[TPS_MANI_01337]	Standardized values for attribute Process.functionClusterAffiliation
[TPS_MANI_01338]	Semantics of SecurityEventReportToSecurityEventDefinitionMapping
[TPS_MANI_01339]	Existence of the SecurityEventReportToSecurityEventDefinitionMapping is motivated by the AUTOSAR methodology
[TPS_MANI_01340]	Semantics of SecurityEventReportInterface
[TPS_MANI_01341]	Security events that are actually reported by a local IdsM
[TPS_MANI_01342]	Semantics of SecurityEventMapping

▽

△

Number	Heading
[TPS_MANI_03235]	Usage of ApSomeipTransformationProps.sessionHandling
[TPS_MANI_03236]	Mapping of ProvidedSomeipServiceInstance to different PPortPrototypes
[TPS_MANI_03237]	Transport Protocol attributes defined for a RequiredSomeipServiceInstance
[TPS_MANI_03238]	Definition of ComMethodGrantDesign.remoteSubject
[TPS_MANI_03239]	Definition of ComEventGrantDesign.remoteSubject
[TPS_MANI_03240]	Modeling of a remote peer in case of TLS-based secure channel
[TPS_MANI_03241]	Modeling of relevant TlsSecureComProps for TlsIamRemoteSubject
[TPS_MANI_03242]	Modeling of a remote peer in case of IPsec-based secure channel
[TPS_MANI_03244]	Modeling of a remote peer in case of a general IP communication
[TPS_MANI_03245]	Definition of ComMethodGrant.remoteSubjects on server side
[TPS_MANI_03246]	Definition of ComMethodGrant.remoteSubjects on client side
[TPS_MANI_03247]	Definition of ComEventGrant.remoteSubjects on provider side
[TPS_MANI_03248]	Definition of ComEventGrant.remoteSubjects on receiver side
[TPS_MANI_03249]	Definition of ComFieldGrant.remoteSubjects on provider side
[TPS_MANI_03250]	Definition of ComFieldGrant.remoteSubjects on client side
[TPS_MANI_03251]	Definition of ComFieldGrantDesign.remoteSubject
[TPS_MANI_03252]	Usage of same End2EndMethodProtectionProps.sourceId in case of Multi-Binding
[TPS_MANI_03253]	Interaction with crypto software
[TPS_MANI_03254]	Modeling of application that uses and modifies a Crypto Key
[TPS_MANI_03255]	Modeling of Key Manager application that manages a Crypto Key that is used by Stack Services
[TPS_MANI_03256]	Modeling of application that accesses a Crypto Certificate
[TPS_MANI_03257]	Modeling of application that accesses a Crypto Provider
[TPS_MANI_03258]	Modeling of application designed as trust-master
[TPS_MANI_03259]	Linking of Crypto Certificate to a Crypto Key Slot
[TPS_MANI_03260]	Semantics of meta-class CryptoModuleInstantiation
[TPS_MANI_03261]	Support of CryptoProviders
[TPS_MANI_03262]	Semantics of CryptoProviderToPortPrototypeMapping
[TPS_MANI_03263]	Assignment of CryptoKeySlots to CryptoProviders
[TPS_MANI_03264]	Semantics of CryptoKeySlotToPortPrototypeMapping
[TPS_MANI_03265]	Support of CryptoCertificates
[TPS_MANI_03266]	Semantics of CryptoCertificateToCryptoKeySlotMapping
[TPS_MANI_03267]	Semantics of CryptoCertificateToPortPrototypeMapping
[TPS_MANI_03268]	Semantics of FunctionalClusterInteractsWithFunctionalClusterMapping
[TPS_MANI_03269]	Semantics of ComCertificateToCryptoCertificateMapping

▽

△

Number	Heading
[TPS_MANI_03270]	Semantics of ComKeyToCryptoKeySlotMapping
[TPS_MANI_03271]	Semantics of ComSecOcToCryptoKeySlotMapping
[TPS_MANI_03272]	Semantics of PersistencyDeploymentToCryptoKeySlotMapping
[TPS_MANI_03273]	Semantics of PersistencyDeploymentElementToCryptoKeySlotMapping
[TPS_MANI_03274]	Configuration of log and trace message source
[TPS_MANI_03275]	Configuration of log and trace message source on design level
[TPS_MANI_03276]	Semantics of PersistencyDeploymentToDltLogChannelMapping
[TPS_MANI_03622]	DDS Transport Protocols are up to the stack implementer
[TPS_MANI_03623]	Usage of checkpointId in application code
[TPS_MANI_03624]	Usage of statusId in application code
[TPS_MANI_03625]	Consistency of HealthChannelExternalReportedStatus.statusId and PhmHealthChannelStatus.statusId
[TPS_MANI_03626]	Consistency of SupervisionCheckpoint.phmCheckpoint and PhmCheckpoint.checkpointId
[TPS_MANI_03627]	No signal-service-translation for methods
[TPS_MANI_03628]	Standardized values of ServiceInterface.category
[TPS_MANI_03629]	Relation of ServiceInstanceToSignalMapping and CommunicationConnector
[TPS_MANI_03630]	Semantics of triggersRecoveryNotification
[TPS_MANI_03631]	Semantics of meta-class PhmHealthChannelRecoveryNotificationInterface
[TPS_MANI_03632]	Semantics of TimeBaseProviderToPersistencyMapping

Table E.33: Added Traceables in R20-11

E.7.2 Changed Traceables in R20-11

Number	Heading
[TPS_MANI_01061]	Requirements on scheduling
[TPS_MANI_01065]	Purpose of PersistencyKeyValueStorageInterface
[TPS_MANI_01067]	Purpose of PersistencyFileStorageInterface
[TPS_MANI_01068]	Semantics of PersistencyFileStorageInterface.maxNumberOfFiles
[TPS_MANI_01073]	Semantics of PortPrototype typed by PersistencyKeyValueStorageInterface
[TPS_MANI_01078]	Semantics of PersistencyPortPrototypeToKeyValueStorageMapping
[TPS_MANI_01079]	Semantics of PersistencyKeyValueStorage
[TPS_MANI_01080]	Semantics of meta-class PersistencyPortPrototypeToFileStorageMapping

▽

△

Number	Heading
[TPS_MANI_01081]	Semantics of PortPrototype typed by PersistencyFileStorageInterface
[TPS_MANI_01106]	Specification of intentions for the receiver of events or field notifiers
[TPS_MANI_01107]	Specification of intentions for the sender of events or field notifiers
[TPS_MANI_01108]	Specification of intentions for the caller of a methods or field setter/getter
[TPS_MANI_01135]	Semantics of PersistencyKeyValueStorageInterface.dataTypeForSerialization
[TPS_MANI_01138]	Semantics of PersistencyKeyValueStorageInterface.dataElement
[TPS_MANI_01139]	Semantics of PersistencyInterface.updateStrategy
[TPS_MANI_01140]	Semantics of PersistencyInterfaceElement.updateStrategy
[TPS_MANI_01142]	Semantics of PersistencyFileElement
[TPS_MANI_01144]	Semantics of PersistencyKeyValuePair
[TPS_MANI_01147]	Semantics of PersistencyDeployment.updateStrategy
[TPS_MANI_01148]	Semantics of PersistencyDeploymentElement.updateStrategy
[TPS_MANI_01149]	Semantics of PersistencyFileStorage.file
[TPS_MANI_01150]	Semantics of PersistencyFileStorage
[TPS_MANI_01155]	PersistencyDeployment.updateStrategy overrides PersistencyInterface.updateStrategy
[TPS_MANI_01156]	PersistencyDeploymentElement.updateStrategy overrides PersistencyDeployment.updateStrategy
[TPS_MANI_01157]	Semantics of updateStrategy on collection level
[TPS_MANI_01159]	Semantics of updateStrategy on element level
[TPS_MANI_01160]	Definition of initial value for PersistencyDataElement
[TPS_MANI_01164]	Semantics of SoftwareCluster.dependsOn
[TPS_MANI_01176]	Standardized value for attribute CppImplementationDataType.typeEmitter
[TPS_MANI_01177]	Semantics of attribute CppImplementationDataType.typeEmitter
[TPS_MANI_01179]	Semantics of PersistencyFileElement.contentUri/PersistencyFile.contentUri vs. PersistencyFileStorage.uri and PersistencyFileElement.fileName/PersistencyFile.fileName
[TPS_MANI_01180]	Collection of data types that requires serialization support
[TPS_MANI_01182]	Value of PersistencyDeploymentElement.updateStrategy overrides PersistencyInterfaceElement.updateStrategy
[TPS_MANI_01187]	Matching pairs of PersistencyFileElement and PersistencyFile
[TPS_MANI_01194]	Semantics of PersistencyInterface.minimumSustainedSize
[TPS_MANI_01196]	Semantics of PersistencyDeployment.minimumSustainedSize
[TPS_MANI_01197]	Semantics of PersistencyDeployment.maximumAllowedSize
[TPS_MANI_01204]	Specification of redundancy of persistent data
[TPS_MANI_01206]	Modeling of redundancy in the context of PersistencyDeployment

▽

△

Number	Heading
[TPS_MANI_01207]	Standardized values of attribute PersistencyRedundancyChecksum.algorithmFamily
[TPS_MANI_01213]	Semantics of meta-class StrongRevisionLabelString
[TPS_MANI_01214]	Semantics of SoftwareCluster.conflictsTo
[TPS_MANI_01215]	Semantics of meta-class SoftwareClusterDependencyFormula
[TPS_MANI_01216]	Semantics of meta-class SoftwareClusterDependencyFormulaPart
[TPS_MANI_01217]	Semantics of meta-class SoftwareClusterDependencyCompareCondition
[TPS_MANI_01263]	Mapping of DiagnosticDataIdentifier or DiagnosticDataElement to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01272]	Duplicate entries in logTraceLogMode
[TPS_MANI_01280]	Semantics of meta-class PhmSupervisionRecoveryNotificationInterface
[TPS_MANI_01284]	Granularity of meta-class RawDataStreamGrantDesign
[TPS_MANI_01285]	Purpose of meta-class RawDataStreamDeployment
[TPS_MANI_01300]	Semantics of reference SoftwarePackageStep.transfer.transfer
[TPS_MANI_03059]	RequiredSomeipServiceInstance.requiredServiceInstanceId
[TPS_MANI_03111]	Mapping between method and operationlocated in a ClientServerInterface
[TPS_MANI_03113]	Mapping between a field and elements of Classic Platform PortInterfaces
[TPS_MANI_03130]	Standardized E2EProfileConfiguration.profileName values
[TPS_MANI_03145]	Description of a function group
[TPS_MANI_03160]	Further configuration options in DltLogChannel
[TPS_MANI_03163]	Network configuration for Log and Trace messages
[TPS_MANI_03165]	Network Interface configuration for DoIP
[TPS_MANI_03194]	Function Group State
[TPS_MANI_03195]	Off state in Function Group
[TPS_MANI_03207]	IPsec Internet Key Exchange protocol configuration
[TPS_MANI_03209]	The meaning of MachineDesign.accessControl
[TPS_MANI_03516]	Status for HealthChannelSupervision
[TPS_MANI_03517]	Evaluation of HealthChannelExternalStatus
[TPS_MANI_03535]	Definition of Time Synchronization interaction
[TPS_MANI_03536]	Time Synchronization interaction in a provider role
[TPS_MANI_03537]	Time Synchronization interaction in a consumer role
[TPS_MANI_03539]	Definition of Time-Base Resources
[TPS_MANI_03541]	Definition of SynchronizedTimeBaseConsumer
[TPS_MANI_03542]	Definition of SynchronizedTimeBaseProvider
[TPS_MANI_03543]	Definition of time sync correction attributes

▽

△

Number	Heading
[TPS_MANI_03546]	Definition of reported health status `RPortPrototype`
[TPS_MANI_03548]	Definition of `TimeSyncPortPrototypeToTimeBaseMapping`
[TPS_MANI_03549]	Usage of `PortPrototype` for the interaction with Time Synchronization
[TPS_MANI_03551]	Definition of Time Base kind
[TPS_MANI_03556]	DDS-RPC Service Binding
[TPS_MANI_03557]	DDS `ClientServerOperation` Binding
[TPS_MANI_03558]	DDS `Field` Binding
[TPS_MANI_03598]	Expected check period of E2E-Protected payload
[TPS_MANI_03599]	Expected update period of E2E-Protected payload
[TPS_MANI_03612]	Sufficient ASIL level of translation software
[TPS_MANI_03617]	Version mapping between `ServiceInterface` and `ServiceInterfaceDeployment`

Table E.34: Changed Traceables in R20-11

E.7.3 Deleted Traceables in R20-11

Number	Heading
[TPS_MAIN_01281]	Usage of meta-class `RecoveryViaApplicationAction`
[TPS_MANI_01015]	Semantics of meta-class `StartupOption`
[TPS_MANI_01059]	Different values of `optionKind` within a `StartupConfig.startupOption`
[TPS_MANI_01069]	Further qualification of properties of `PortPrototype`s typed by `PersistencyKeyValueDatabaseInterface`s
[TPS_MANI_01141]	Semantics of `PersistencyFileProxyInterface.updateStrategy`
[TPS_MANI_01143]	Semantics of `PersistencyFileProxy.updateStrategy`
[TPS_MANI_01151]	Semantics of `PersistencyFileArray.updateStrategy`
[TPS_MANI_01152]	Semantics of `PersistencyFile.updateStrategy`
[TPS_MANI_01154]	`PersistencyFileArray.updateStrategy` overrides `PersistencyFileProxyInterface.updateStrategy`
[TPS_MANI_01158]	`PersistencyFile.updateStrategy` overrides `PersistencyFileArray.updateStrategy`
[TPS_MANI_01163]	Impact of values of `category` on the semantics of `SoftwareCluster`
[TPS_MANI_01183]	`PersistencyFile.updateStrategy` overrides `PersistencyFileProxy.updateStrategy`
[TPS_MANI_01195]	Semantics of `PersistencyFileProxyInterface.minimumSustainedSize`
[TPS_MANI_01224]	Actions taken after installation of a `SoftwarePackage`
[TPS_MANI_01264]	Mapping of `DiagnosticServiceInstance` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01266]	Mapping of `DiagnosticServiceInstance` for upload/download to `PortPrototype`(s) on the *AUTOSAR adaptive platform*

▽

△

Number	Heading
[TPS_MANI_01267]	Semantics of attribute SoftwareClusterDesign.dependsOn
[TPS_MANI_01268]	Semantics of attribute SoftwareClusterDesign.conflictsTo
[TPS_MANI_01283]	Semantics of meta-class RawDataStreamInterface
[TPS_MANI_01286]	Semantics of attribute RawDataStreamMethodDeployment.callTimeout
[TPS_MANI_03065]	Hardware resources of the machine
[TPS_MANI_03161]	Log and Trace configuration options in the Service Instance Manifest
[TPS_MANI_03518]	PhmLogicalExpression definition
[TPS_MANI_03519]	PhmRule definition
[TPS_MANI_03520]	Execution of PhmActionList with actionListExecution=triggeredOnEvaluation
[TPS_MANI_03521]	Execution of PhmActionList with actionListExecution=triggeredOnChange
[TPS_MANI_03522]	Definition of actions for application software
[TPS_MANI_03523]	Definition of actions for Platform Instance
[TPS_MANI_03524]	Definition of actions for Watchdog
[TPS_MANI_03538]	Time Synchronization interaction with a local Time Base
[TPS_MANI_03540]	Definition of PureLocalTimeBase
[TPS_MANI_03559]	Definition of DdsProvidedServiceInstance.methodQosProps
[TPS_MANI_03560]	qosProfile of DdsProvidedServiceInstance.methodQosProps is optional
[TPS_MANI_03563]	Definition of DdsProvidedServiceInstance.fieldGetSetQosProps
[TPS_MANI_03564]	qosProfile of DdsProvidedServiceInstance.fieldGetSetQosProps is optional
[TPS_MANI_03565]	Definition of DdsRequiredServiceInstance.methodQosProps
[TPS_MANI_03566]	qosProfile of DdsRequiredServiceInstance.methodQosProps is optional
[TPS_MANI_03569]	Definition of DdsRequiredServiceInstance.fieldGetSetQosProps
[TPS_MANI_03570]	qosProfile of DdsRequiredServiceInstance.fieldGetSetQosProps is optional
[TPS_MANI_03571]	transportPlugin for DdsProvidedServiceInstance
[TPS_MANI_03572]	transportPlugin for DdsRequiredServiceInstance

Table E.35: Deleted Traceables in R20-11

E.7.4 Added Constraints in R20-11

Number	Heading
[constr_1743]	CppImplementationDataType.headerFile vs. CppImplementationDataType.typeEmitter
[constr_1744]	Definition of process state In the context of the ExecutionDependency
[constr_1746]	Mutual exclusive existence of PersistencyInterface.redundancy and PersistencyInterface.redundancyHandling
[constr_1747]	Completeness of the SoftwareCluster.version
[constr_1748]	Existence of references TlvDataIdDefinition.tlvArgument, TlvDataIdDefinition.tlvRecordElement, and TlvDataIdDefinition.tlvSubElement
[constr_1751]	Value of PersistencyRedundancyMOutOfN.n and PersistencyRedundancyMOutOfN.m
[constr_1764]	Counterpart of PhmCheckpoint
[constr_1765]	Diagnostic Services eligible for DiagnosticServiceGenericMapping
[constr_1769]	Existence of ProcessArgument.argument
[constr_1770]	Value of ProvidedSomeipServiceInstance.serviceInstanceId
[constr_1784]	Restriction for the reference to UploadableExclusivePackageElement
[constr_1785]	Restriction regarding the reference into another SoftwareCluster
[constr_1786]	Restriction to use functionGroup in terms of SoftwareCluster
[constr_1787]	Restricted use of function groups in the context of a SoftwareCluster
[constr_1788]	Restriction to SoftwareCluster of category PLATFORM_CORE
[constr_1789]	Scope of machine function group
[constr_1809]	Global supervision restricted to one function group
[constr_3563]	Mandatory topic name values
[constr_3564]	Consistency between DDS Service Interface Deployment and Provided DDS Service Instance
[constr_3565]	Consistency between DDS Service Interface Deployment and Required DDS Service Instance
[constr_3568]	No support for cross PlatformHealthManagementContribution references
[constr_3569]	Applicability of attribute invalidValue on CppImplementationDataType of category TYPE_REFERENCE
[constr_3612]	Multiplicity of references recoveryNotification, recoveryAction, and process at RecoveryNotificationToPPortPrototypeMapping
[constr_3613]	Reference to a PhmSupervisionRecoveryNotificationInterface in the context of a HealthChannelSupervision
[constr_3614]	Reference to a PhmHealthChannelRecoveryNotificationInterface in the context of a HealthChannelExternalStatus
[constr_3619]	Mandatory references of TimeBaseProviderToPersistencyMapping
[constr_5115]	Search for a specific SOME/IP ServiceInstance and for all SOME/IP ServiceInstances over the same RPortPrototype
[constr_5155]	SomeipServiceInstanceToMachineMapping only supports a single Address Family

▽

△

Number	Heading
[constr_5156]	SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_5161]	RequiredSomeipServiceInstance that is mapped by a SomeipServiceInstanceToMachineMapping without a configured tcpPort and udpPort
[constr_5227]	Mandatory elements of UdpNmCluster
[constr_5228]	Partial Networking timing constraint
[constr_5230]	Existence of attribute E2EProfileCompatibilityProps.transitToInvalidExtended is mandatory for each E2EProfileConfiguration
[constr_5238]	CryptoKeySlotAllowedModification.restrictUpdate and the relationship to maxNumberOfAllowedUpdates
[constr_5239]	Predefined values for CryptoKeySlotContentAllowedUsage.allowedKeyslotUsage
[constr_5240]	Restriction applicable for CryptoProviderToPortPrototypeMapping.portPrototype
[constr_5241]	Restriction applicable for CryptoKeySlotToPortPrototypeMapping.portPrototype
[constr_5242]	Restriction applicable for CryptoCertificateToPortPrototypeMapping.portPrototype
[constr_5243]	Restriction of LogAndTraceInstantiation.dltEcuId attribute value
[constr_10002]	Only one mapping per PortPrototype
[constr_10003]	Restriction for the existence of DiagnosticServiceDataIdentifierPortMapping.diagnosticDataIdentifier vs. DiagnosticServiceDataIdentifierPortMapping.diagnosticDataElement
[constr_10004]	Consistency of DiagnosticServiceGenericMapping for PortPrototype typed by DiagnosticDataIdentifierGenericInterface
[constr_10007]	Existence of ProcessExecutionError.executionError
[constr_10008]	Value of ProcessExecutionError.executionError
[constr_10010]	Usage of attribute category in a SoftwareClusterDependencyFormula
[constr_10011]	Definition of sub-software-cluster
[constr_10021]	Existence of IdsmModuleInstantiation
[constr_10022]	Restriction for SecurityEventMapping.process.securityEvent.id w.r.t SecurityEventMapping.id
[constr_10023]	Mandatory content of any functionGroup

Table E.36: Added Constraints in R20-11

E.7.5 Changed Constraints in R20-11

Number	Heading
[constr_1490]	Allowed value for Executable.category if ProcessToMachineMapping references a NonOsModuleInstantiation
[constr_1500]	Target SwcServiceDependency of DiagnosticEventPortMapping.swcServiceDependencyInExecutable
[constr_1507]	PortInterfaceToDataTypeMapping is only applicable to ServiceInterface or PersistencyKeyValueStorageInterface
[constr_1555]	Restriction applicable for PersistencyPortPrototypeToKeyValueStorageMapping.portPrototype
[constr_1556]	Restriction applicable for PersistencyPortPrototypeToFileStorageMapping.portPrototype
[constr_1564]	Existence of SoftwareCluster.diagnosticAddress
[constr_1566]	Usage of SoftwareCluster.containedARElement
[constr_1568]	Existence of SoftwareCluster.diagnosticExtract
[constr_1581]	Value of fileElement.fileName
[constr_1589]	Value of file.fileName
[constr_1613]	File name of matching pairs of PersistencyFileElement and PersistencyFile
[constr_1659]	Restriction for the usage of CppImplementationDataTypeElementQualifier.inplace
[constr_1666]	References from PersistencyPortPrototypeToKeyValueStorageMapping to PersistencyKeyValueStorage
[constr_1667]	References from PersistencyPortPrototypeToFileStorageMapping to PersistencyFileStorage
[constr_1668]	Allowed combinations of PersistencyRedundancyChecksum.length and algorithmFamily
[constr_1673]	Existence of attributes hasGetter, hasSetter, and hasNotifier
[constr_1710]	Consistency of values of attributes PersistencyInterface.redundancy and PersistencyRedundancyHandling.scope
[constr_1729]	Qualified combinations of PortPrototypes and PhmSupervisionRecoveryNotificationInterface / PhmHealthChannelRecoveryNotificationInterface on State Management software level
[constr_3305]	Value of attribute SomeipEventDeployment.eventId shall be unique
[constr_3306]	Value of attribute methodId shall be unique per SomeipServiceInterfaceDeployment
[constr_3356]	Restriction in usage of ApSomeipTransformationProps.alignment
[constr_3414]	Allowed usage of PlatformModuleEthernetEndpointConfiguration attributes
[constr_3421]	Fibex elements applicable for a System of category MACHINE_DESIGN_EXTRACT
[constr_3426]	The logTraceFilePath is mandatory in case that logTraceLogMode is set to file
[constr_3427]	The logTraceFilePath is only relevant if logTraceLogMode is set to file
[constr_3493]	Applicable attributes for standardized E2E Profiles

▽

△

Number	Heading
[constr_3552]	Full mapping of target event
[constr_3554]	E2E protection configuration check
[constr_5052]	ProvidedSomeipServiceInstances of the same serviceInterface on one Machine

Table E.37: Changed Constraints in R20-11

E.7.6 Deleted Constraints in R20-11

Number	Heading
[constr_1481]	Usage of DataPrototypeInServiceInterfaceRef in the *AUTOSAR adaptive platform*
[constr_1497]	Attribute optionKind set to commandLineSimpleForm
[constr_1498]	Attribute optionKind set to commandLineShortForm or commandLineLongForm
[constr_1524]	Standardized values of PersistencyFileProxyInterface.category
[constr_1534]	Existence of DiagnosticSoftwareClusterProps
[constr_1542]	No nested definition of SoftwareCluster
[constr_1561]	Existence of SoftwareClusterDesign.subSoftwareCluster and SoftwareClusterDesign.dependsOn.dependentSoftwareClusterDesign
[constr_1563]	Standardized values of SoftwareClusterDesign.category and SoftwareCluster.category
[constr_1565]	Existence of SoftwareCluster.subSoftwareCluster
[constr_1567]	Existence of SoftwareCluster.subSoftwareCluster and SoftwareCluster.dependsOn/conflictsTo
[constr_1615]	Existence of attribute SomeipDataPrototypeTransformationProps.someipTransformationProps.sessionHandling
[constr_1687]	Definition of machine state
[constr_1703]	Target SwcServiceDependency of DiagnosticGenericUdsPortMapping.swcServiceDependencyInExecutable
[constr_1704]	Target SwcServiceDependency of DiagnosticUploadDownloadPortMapping.swcServiceDependencyInExecutable
[constr_1705]	Target of reference SoftwareActivationDependencyCompareCondition.softwareActivationDependency
[constr_1709]	Applicability of attribute PersistencyRedundancyEnum.redundantPerKey
[constr_1730]	Restriction regarding the modeling of the PhmRecoveryActionInterface.recovery
[constr_3296]	Transport Protocol attributes defined for a RequiredSomeipServiceInstance
[constr_3297]	SomeipServiceInstanceToMachineMapping only supports a single Address Family
[constr_3307]	SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances

▽

△

Number	Heading
[constr_3412]	`OsModuleInstantiation` shall have at least one `ResourceGroup`
[constr_3527]	`PhmLogicalExpression` referenced by one `PhmRule`
[constr_3543]	At least one `transportPlugin` definition required for each `DdsProvidedServiceInstance`
[constr_3544]	At least one `transportPlugin` definition required for each `DdsRequiredServiceInstance`
[constr_3556]	Unique transport layer mapping

Table E.38: Deleted Constraints in R20-11

F Splitable Elements in the Scope of this Document

This chapter contains a table of all model elements stereotyped ≪atpSplitable≫ in the scope of this document.

Each entry in the table consists of the identification of the specific model element itself and the applicable value of the tagged value atp.Splitkey.

For more information about the concept of splitable model elements and how these shall be treated please refer to [6].

Name of splitable element	*Splitkey*
AdaptiveApplicationSwComponentType.internalBehavior	internalBehavior.shortName, internal Behavior.variationPoint.shortLabel
CryptoProvider.keySlot	keySlot.shortName
DiagnosticClearConditionGroup.clearCondition	clearCondition
DiagnosticClearConditionPortMapping.process	process
DiagnosticIndicatorPortMapping.process	process
DiagnosticMemoryDestinationPortMapping.process	process
DiagnosticSecurityLevelPortMapping.process	process
DiagnosticServiceDataIdentifierPortMapping.process	process
DiagnosticServiceGenericMapping.process	process
IamModuleInstantiation.grant	grant
IdsmModuleInstantiation.reportableSecurityEvent	reportableSecurityEvent
InterfaceMappingSet.interfaceMapping	interfaceMapping.shortName, interface Mapping.variationPoint.shortLabel
Machine.environmentVariable	environmentVariable, environmentVariable.variation Point.shortLabel
Machine.moduleInstantiation	moduleInstantiation.shortName
Machine.secureCommunicationDeployment	secureCommunicationDeployment.shortName
PlatformHealthManagementContribution.checkpoint	checkpoint.shortName
PlatformHealthManagementContribution.globalSupervision	globalSupervision.shortName
PlatformHealthManagementContribution.healthChannel	healthChannel.shortName
PlatformHealthManagementContribution.localSupervision	localSupervision.shortName
Process.securityEvent	securityEvent
ServiceInstanceToPortPrototypeMapping.process	process
SoftwareCluster.conflictsTo	conflictsTo
SoftwareCluster.containedARElement	containedARElement
SoftwareCluster.containedPackageElement	containedPackageElement
SoftwareCluster.dependsOn	dependsOn
SoftwareCluster.diagnosticAddress	diagnosticAddress
SoftwareCluster.moduleInstantiation	moduleInstantiation
SoftwareClusterDesign.containedProcess	containedProcess
SoftwareClusterDesign.diagnosticAddress	diagnosticAddress
SoftwareClusterDesign.diagnosticContribution	diagnosticContribution
SoftwareClusterDesign.requiredARElement	requiredARElement
SoftwareClusterDesign.requiredFibexElement	requiredFibexElement

▽

△

Name of splitable element	*Splitkey*
SoftwareClusterDesign.requiredPackageElement	requiredPackageElement
SoftwareClusterDesign.subSoftwareCluster	subSoftwareCluster

Table F.1: Usage of splitable elements

G Variation Points in the Scope of this Document

This chapter contains a table of all model elements stereotyped ≪atpVariation≫ in the scope of this document.

Each entry in the table consists of the identification of the model element itself and the applicable value of the tagged value vh.latestBindingTime.

For more information about the concept of variation points and how model elements that contain variation points shall be treated please refer to [6].

Variation Point	*Latest Binding Time*
AdaptiveApplicationSwComponentType.internalBehavior	preCompileTime
CppImplementationDataType.arraySize	preCompileTime
IdsPlatformInstantiation.timeBase	systemDesignTime
InterfaceMappingSet.interfaceMapping	systemDesignTime
ServiceInterface.event	blueprintDerivationTime
ServiceInterface.field	blueprintDerivationTime
ServiceInterface.method	blueprintDerivationTime

Table G.1: Usage of variation points